Linear Dependent Types and Relative Completeness

Ugo Dal Lago, Marco Gaboardi

Logical Methods in Computer Science
Vol. 8(4:11)2012, pp. 1–44
www.lmcs-online.org

Submitted Dec. 29, 2011
Published Oct. 23, 2012

LINEAR DEPENDENT TYPES AND RELATIVE COMPLETENESS *

UGO DAL LAGO [a] AND MARCO GABOARDI [b]

[a] Dipartimento di Scienze dell'Informazione – Università di Bologna, EPI FOCUS – INRIA Sophia Antipolis
e-mail address: dallago@cs.unibo.it

[b] Dipartimento di Scienze dell'Informazione – Università di Bologna, Computer and Information Science Department – University of Pennsylvania, EPI FOCUS – INRIA Sophia Antipolis
e-mail address: gaboardi@cs.unibo.it

Abstract. A system of linear dependent types for the λ-calculus with full higher-order recursion, called $\mathsf{d}\ell\mathsf{PCF}$, is introduced and proved sound and relatively complete. Completeness holds in a strong sense: $\mathsf{d}\ell\mathsf{PCF}$ is not only able to precisely capture the functional behavior of PCF programs (i.e. how the output relates to the input) but also some of their intensional properties, namely the complexity of evaluating them with Krivine's Machine. $\mathsf{d}\ell\mathsf{PCF}$ is designed around dependent types and linear logic and is parametrized on the underlying language of index terms, which can be tuned so as to sacrifice completeness for tractability.

1. Introduction

Type systems are powerful tools in the design of programming languages. While they have been employed traditionally to guarantee weak properties of programs (e.g. "well-typed programs cannot go wrong"), it is becoming more and more evident that they can be useful when stronger properties are needed, such as security [33, 32], termination [6], monadic temporal properties [26] or resource bounds [25, 11].

One key advantage of type systems seen as formal methods is their simplicity and their close relationship with programs — checking whether a program has a type or even inferring the (most general) type of a program is often decidable. The price to pay is the incompleteness of most type systems: there are programs satisfying the property at hand which cannot be given a type. This is in contrast with other formal methods, like program logics [2] where completeness is always a desirable feature, although it only holds relatively

1998 ACM Subject Classification: F.3.2, F.3.1.

Key words and phrases: Resource Consumption, Linear Logic, Dependent Types, Implicit Computational Complexity, Relative Completeness.

* This work is partially supported by the INRIA ARC project "ETERNAL". This is a revised and extended version of a paper with the same title which has appeared in the proceedings of LICS 2011.

[b] Marco Gaboardi was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 272487.

DOI:10.2168/LMCS-8(4:11)2012

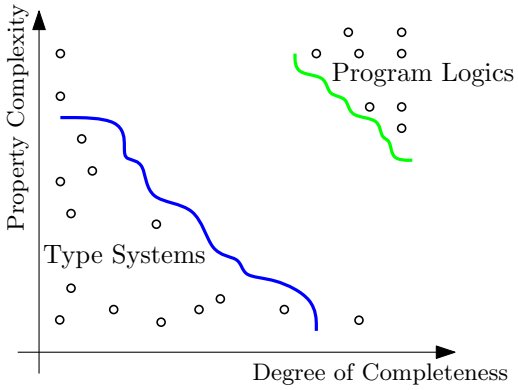


Figure 1: Type Systems and Program Logics

to an oracle. Graphically, the situation is similar to the one in Figure 1: type systems are bound to be in the lower left corner of the diagram, where both the degree of completeness and the complexity of the property under consideration is low; program logics, on the other hand, are confined to the upper-right corner, where checking for derivability is almost always undecidable.

One specific research field in which the just-described scenario manifests itself is implicit computational complexity, in which one aims at defining characterizations of complexity classes by programming languages and logical systems. Many type systems have been introduced capturing, for instance, the polynomial time computable functions [23, 5, 4]. All of them, under mild assumptions, can be employed as tools to certify programs as asymptotically time efficient. However, a tiny slice of the polytime *programs* are generally typable, since the underlying complexity class **FP** is only characterized in a purely extensional sense — for every function in **FP** there is *at least one* typable program computing it.

The main contribution of this paper is a *type system* for the λ-calculus with full recursion, called $\mathsf{d}\ell\mathsf{PCF}$, which is sound *and complete*. Types of $\mathsf{d}\ell\mathsf{PCF}$ are obtained, in the spirit of DML [36, 35], by decorating types of ordinary PCF [31, 21] with *index terms*. These are first-order terms freely generated from variables, function symbols and a few more term constructs. They are indicated with metavariables like $\mathrm{I}, \mathrm{J}, \mathrm{K}$. Type decoration reflects the standard decomposition of types into *linear types* (as suggested by linear logic [18]), and is inspired by recent works on the expressivity of bounded logics [13].

Index terms and linear types permit to describe program properties with a fine granularity. More precisely, $\mathsf{d}\ell\mathsf{PCF}$ enjoys the following two properties:

- *Soundness*: if t is a program and $\vdash_{\mathrm{K}} t : \mathtt{Nat}[\mathrm{I}, \mathrm{J}]$, then t evaluates to a natural number which lies between I and J and this evaluation takes at most $(\mathrm{K}+1)\cdot|t|$ steps;
- *Completeness*: if t is typable in PCF and evaluates to a natural number n in m steps, then $\vdash_{\mathrm{I}} t : \mathtt{Nat}[\mathbf{n}, \mathbf{n}]$, where $\mathrm{I} \leqslant m$.

Completeness of $\mathsf{d}\ell\mathsf{PCF}$ holds not only for programs (i.e. terms of ground types) but also for functions on the natural numbers (see Section 5.3 for further details). Moreover, typing judgments tell us something about the functional behavior of programs but also about their non-functional one, namely the number of steps needed to evaluate the term in Krivine's Abstract Machine.

As the title of this paper suggests, completeness of $\mathsf{d}\ell\mathsf{PCF}$ holds in a relative sense. Indeed, the behavior of programs can be precisely captured only in presence of a complete

oracle for the truth of certain assumptions in typing rules. This is exactly what happens in program logics such as Floyd-Hoare's logic, where all true partial correctness assertions can be derived *provided* one is allowed to use all true sentences of first order arithmetic as axioms [10]. In $\mathsf{d}\ell\mathsf{PCF}$, those assumptions take the form of (in)equalities between index terms, to be verified when function symbols are interpreted as partial functions on natural numbers according to an equational program $\mathcal{E}$. Actually, the whole of $\mathsf{d}\ell\mathsf{PCF}$ is *parametrized* on such an $\mathcal{E}$, but while soundness holds independently of the specific $\mathcal{E}$, completeness, as is to be expected, holds only if $\mathcal{E}$ is sufficiently powerful to encode all total computable functions (i.e. if $\mathcal{E}$ is universal). In other words, $\mathsf{d}\ell\mathsf{PCF}$ can be claimed to be not *a* type system, but *a family of* type systems obtained by taking a specific $\mathcal{E}$ as the underlying "logic" of index terms. The simpler $\mathcal{E}$, the easier type checking and type inference are; the more complex $\mathcal{E}$, the larger the class of captured programs.

The design of $\mathsf{d}\ell\mathsf{PCF}$ has been very much influenced by linear logic [18], and in particular by systems of indexed and bounded linear logic [19, 13], which have been recently shown to subsume other ICC systems as for the class of programs they capture [13]. One of the many ways to "read" $\mathsf{d}\ell\mathsf{PCF}$ is as a variation on the theme of BLL [19] obtained by generalizing polynomials to arbitrary functions. The idea of going beyond a restricted, fixed class of bounds comes from Xi's work on DML [36, 35]. Cost recurrences for first order DML programs have been studied [20]. No similar completeness results for dependent types are known, however.

2. Types and Program Properties: An Informal Account

Consider the following program:

$$\mathtt{dbl} = \mathtt{fix}\ f.\lambda x.\ \mathtt{ifz}\ x\ \mathtt{then}\ \underline{\mathtt{0}}\ \mathtt{else}\ \mathtt{s}(\mathtt{s}(f(\mathtt{p}(x)))).$$

In a monomorphic, traditionally designed type system like PCF [31, 21], the term `dbl` receives type $\mathtt{Nat} \to \mathtt{Nat}$. As a consequence, `dbl` computes a function on natural numbers without "going wrong": it takes in input a natural number, and produces in output another natural number (if any). The type $\mathtt{Nat} \to \mathtt{Nat}$, however, does not give any information about *which* specific function on the natural numbers `dbl` computes. Indeed, in PCF (and in most real-world programming languages) any program computing a function on natural numbers, being it for instance the identity function or (a unary version of) the Ackermann function, can be typed by $\mathtt{Nat} \to \mathtt{Nat}$.

Some modern type systems allow one to construct and use types like $\tau = \mathtt{Nat}[a] \to \mathtt{Nat}[\mathbf{2} \times a]$, which tell not only what set or domain (the interpretation of) the term belongs to, but also which specific element of the domain the term actually denotes. The type τ, for example, could be attributed only to those programs computing the function $n \mapsto 2 \times n$, including `dbl`. Types of this form can be constructed in dependent and sized type theories [36, 6]. The type system $\mathsf{d}\ell\mathsf{PCF}$ introduced in this paper offers this possibility, too. But, as a first contribution, it further allows to specify *imprecise* types, like $\mathtt{Nat}[\mathbf{5}, \mathbf{8}]$, which stands for the type of those natural numbers between 5 and 8 (included).

A property of programs which is completely ignored by ordinary type systems is termination, at least if full recursion is in the underlying language. Typing a term t with $\mathtt{Nat} \to \mathtt{Nat}$ does not guarantee that t, when applied to a natural number, terminates. In PCF this is even worse: t could possibly diverge *itself*! Consider, as another example, a

slight modification of dbl, namely

$$\texttt{omega} = \texttt{fix}\ f.\lambda x.\ \texttt{ifz}\ x\ \texttt{then}\ \underline{0}\ \texttt{else}\ \texttt{s}(\texttt{s}(f(x))).$$

It behaves as dbl when fed with 0, but it diverges when it receives a positive natural number as an argument. But look: omega is not so different from dbl. Indeed, the second can be obtained from the first by feeding not x but $\texttt{p}(x)$ to f. And any type systems in which dbl and omega are somehow recognized as being fundamentally different must be able to detect the presence of p in dbl and deduct termination from it. Indeed, sized types [6] and dependent types [34] are able to do so.

Going further, we could ask the type system to be able not only to guarantee termination, but also to somehow evaluate the time or space consumption of programs. For example, we could be interested in knowing that dbl takes a polynomial number of steps to be evaluated on any natural number. This cannot be achieved neither using classical type systems nor using systems of sized types, at least when traditionally formulated. However, some type systems able to control the complexity of programs exist. Good examples are type systems for amortized analysis [25, 22] or those using ideas from linear logic [5, 4]. In those type systems, typing judgements carry, besides the usual type information, some additional information about the resource consumption of the underlying program. As an example, dbl could be given a type as follows

$$\vdash_{\mathrm{I}} \texttt{dbl} : \texttt{Nat} \to \texttt{Nat}$$

where I is some cost information for dbl. This way, building a type derivation and inferring resource consumption can be done at the same time.

The type system $\mathsf{d}\ell\mathsf{PCF}$ we propose in this paper makes some further steps in this direction. First of all, it combines some of the ideas presented above with the ones of bounded linear logic. BLL allows one to explicitly count the number of times functions use their arguments (in rough notation, $!_{\mathbf{m}}\sigma \multimap \tau$ says that the argument of type σ is used m times). This permits to extract natural cost functions from type derivations. The cost of evaluating a term will be measured by counting how many times function arguments need to be copied during evaluation. Making this information explicit in types permits to compute the cost step by step during the type derivation process. By the way, previous works by the first author [12] show that this way of attributing a cost to (proofs seen as) programs is sound and precise as a way to measure their time complexity. Intuitively, typing judgements in $\mathsf{d}\ell\mathsf{PCF}$ can be thought as:

$$\vdash_{\mathrm{J}} t\ : !_{\mathbf{m}}\ \texttt{Nat}[a] \multimap \texttt{Nat}[\mathrm{I}].$$

where I and J can be derived while building a type derivation, exploiting the information carried by the modalities. In fact, the quantitative information in $!_{\mathbf{m}}$ allows to statically determine the number of times any subterm will be copied during evaluation. But this is not sufficient: analogously to what happens in BLL, $\mathsf{d}\ell\mathsf{PCF}$ makes types more parametric. A rough type as $!_{\mathbf{n}}\sigma \multimap \tau$ is replaced by the more parametric type $[a < \mathbf{n}] \cdot \sigma \multimap \tau$, which tells us that the argument will be used n times, and each instance has type σ *where, however* the variable a is instantiated with a value less than n. This allows to type each copy of the argument differently but uniformly, since all instances of σ have the same PCF skeleton. This form of *uniform linear dependence* is actually crucial in obtaining the result which makes $\mathsf{d}\ell\mathsf{PCF}$ different from similar type systems, namely completeness.

Finally, as already stressed in the Introduction, dℓPCF is also parametric in the class of functions (in the form of an equational program $\mathcal{E}$) that can be used to reason about types and costs. This permits to tune the type system, as described in Section 6 below.

Anticipating on the next section, we can say that `dbl` can be typed as follows in dℓPCF:

$$\vdash_a^{\mathcal{E}} \texttt{dbl} : [b < a + \mathbf{1}] \cdot \texttt{Nat}[a] \multimap \texttt{Nat}[\mathbf{2} \times a].$$

This tells us that the argument will be used $a + 1$ times by `dbl`, and that the cost of evaluation will be itself proportional to a.

3. dℓPCF

In this section, the language of programs and the type system dℓPCF for it will be introduced formally. Some of their basic properties will be described. The type system dℓPCF is based on the notion of an index term whose semantics, in turn, is defined by an equational program. As a consequence, all these notions must be properly introduced and are the subject of Section 3.1 below.

3.1. Index Terms and Equational Programs.

Syntactically, index terms are built either from function symbols from a given signature or by applying any of two special term constructs.

Formally, a *signature* Σ is a pair $(\mathcal{S}, \alpha)$ where $\mathcal{S}$ is a finite set of *function symbols* and $\alpha : \mathcal{S} \to \mathbb{N}$ assigns an *arity* to every function symbol. Index terms on a given signature $\Sigma = (\mathcal{S}, \alpha)$ are generated by the following grammar:

$$\mathrm{I}, \mathrm{J}, \mathrm{K} ::= a \;\mid\; \mathtt{f}(\mathrm{I}_1, \ldots, \mathrm{I}_{\alpha(\mathtt{f})}) \;\mid\; \sum_{a<\mathrm{I}} \mathrm{J} \;\mid\; \bigcirc\!\!\!\!\!\!\triangle_{a}^{\mathrm{I},\mathrm{J}} \mathrm{K},$$

where $\mathtt{f} \in \mathcal{S}$ and a is a variable drawn from a set $\mathcal{V}$ of *index variables*. We assume the symbols $\mathbf{0}$, $\mathbf{1}$ (with arity 0) and $+$, $\dot{-}$ (with arity 2) are always part of Σ. An index term in the form $\sum_{a<\mathrm{I}} \mathrm{J}$ is a *bounded sum*, while one in the form $\bigcirc\!\!\!\!\!\!\triangle_{a}^{\mathrm{I},\mathrm{J}} \mathrm{K}$ is a *forest cardinality*. For every natural number n, the index term $\mathbf{n}$ is just

$$\underbrace{\mathbf{1} + \mathbf{1} + \ldots + \mathbf{1}}_{n \text{ times}}.$$

Index terms are meant to denote natural numbers, possibly depending on the (unknown) values of variables. Variables can be instantiated with other index terms, e.g. $\mathrm{I}\{\mathrm{J}/a\}$. So, index terms can also act as first order functions. What is the meaning of the function symbols from Σ? It is the one induced by an equational program $\mathcal{E}$. Formally, an *equational program* $\mathcal{E}$ over a signature Σ and a set of variables $\mathcal{V}$ is a set of equations in the form $t = s$ where both t and s are terms in the free algebra $\mathcal{O}(\Sigma, \mathcal{V})$ over Σ and $\mathcal{V}$. We are interested in equational programs guaranteeing that, whenever symbols in Σ are interpreted as partial functions over $\mathbb{N}$ and $\mathbf{0}$, $\mathbf{1}$, $+$ and $\dot{-}$ are interpreted in the usual way, the semantics of any function symbol $\mathtt{f}$ can be uniquely determined from $\mathcal{E}$. This can be guaranteed by, for example, taking $\mathcal{E}$ as an Herbrand-Gödel scheme [30] or as an orthogonal constructor term rewriting system [3]. One may wonder why the definition of index terms is parametric on Σ and $\mathcal{E}$. As we will see in Section 6, being parametric this way allows us to tune our concrete type system from a highly undecidable but truly powerful machinery down to a tractable but less expressive formal system. An example of an equational program over

the signature Σ consisting of three function symbols $\mathtt{gt}$, $\mathtt{add}$ and $\mathtt{mult}$ of arity two is the following sequence of equations:

$$\begin{aligned}
\mathtt{gt}(\mathbf{0}, b) &= \mathbf{0};\\
\mathtt{gt}(a+\mathbf{1}, \mathbf{0}) &= \mathbf{1};\\
\mathtt{gt}(a+\mathbf{1}, b+\mathbf{1}) &= \mathtt{gt}(a, b);\\
\mathtt{add}(\mathbf{0}, b) &= b;\\
\mathtt{add}(a+\mathbf{1}, b) &= \mathtt{add}(a, b)+\mathbf{1};\\
\mathtt{mult}(\mathbf{0}, b) &= \mathbf{0};\\
\mathtt{mult}(a+\mathbf{1}, b) &= \mathtt{add}(b, \mathtt{mult}(a, b)).
\end{aligned}$$

What about the meaning of bounded sums and forest cardinalities? The first is very intuitive: the value of $\sum_{a<\mathrm{I}} \mathrm{J}$ is simply the sum of all possible values of J with a taking the values from 0 up to I, excluded. Forest cardinalities, on the other hand, require some more effort to be described. Informally, $\bigtriangleup_a^{\mathrm{I,J}}\,\mathrm{K}$ is an index term denoting the number of nodes in a forest composed of J trees described using K. All the nodes in the forest are (uniquely) identified by natural numbers. These are obtained by consecutively visiting each tree in pre-order, starting from I. The term K has the role of describing the number of children of each forest node n by properly instantiating the variable a, e.g the number of children of the root (of the leftmost tree in the forest) is $\mathrm{K}\{\mathbf{0}/a\}$. More formally, the meaning of a forest cardinality is defined by the following two equations:

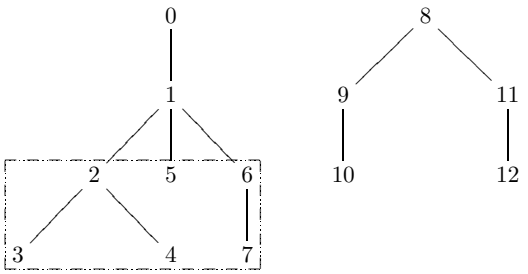
$$\bigtriangleup_a^{\mathrm{I},\mathbf{0}}\,\mathrm{K} = \mathbf{0}; \tag{3.1}$$

$$\bigtriangleup_a^{\mathrm{I},\mathrm{J}+\mathbf{1}}\,\mathrm{K} = \left(\bigtriangleup_a^{\mathrm{I},\mathrm{J}}\,\mathrm{K}\right) + \mathbf{1} + \left(\bigtriangleup_a^{\mathrm{I}+\mathbf{1}+\bigtriangleup_a^{\mathrm{I},\mathrm{J}}\,\mathrm{K},\mathrm{K}\{\mathrm{I}+\bigtriangleup_a^{\mathrm{I},\mathrm{J}}\,\mathrm{K}/a\}}\,\mathrm{K}\right). \tag{3.2}$$

Equation (3.1) says that a forest of 0 trees contains no nodes. Equation (3.2) tells us that a forest of $\mathrm{J}+1$ trees contains:

- the nodes in the first J trees;
- and the nodes in the last tree, which are just one plus the nodes in the immediate subtrees of the root, considered themselves as a forest.

To better understand forest cardinalities, consider the following forest comprising two trees:



and consider an index term K with a free index variable a such that $\mathrm{K}\{\mathbf{n}/a\} = 3$ for $n = 1$; $\mathrm{K}\{\mathbf{n}/a\} = 2$ for $n \in \{2, 8\}$; $\mathrm{K}\{\mathbf{n}/a\} = 1$ when $n \in \{0, 6, 9, 11\}$; and $\mathrm{K}\{\mathbf{n}/a\} = 0$ when $n \in \{3, 4, 7, 10, 12\}$. That is, K describes the number of children of each node in the forest. Then $\bigtriangleup_a^{\mathbf{0},\mathbf{2}}\,\mathrm{K} = \mathbf{13}$ since it takes into account the entire forest; $\bigtriangleup_a^{\mathbf{0},\mathbf{1}}\,\mathrm{K} = \mathbf{8}$ since it takes

into account only the leftmost tree; $\bigtriangleup_a^{\mathbf{8},\mathbf{1}} \mathrm{K} = \mathbf{5}$ since it takes into account only the second tree of the forest; finally, $\bigtriangleup_a^{\mathbf{2},\mathbf{3}} \mathrm{K} = \mathbf{6}$ since it takes into account only the three trees (as a forest) in the dashed rectangle.

One may wonder what is the role of forest cardinalities in the type system. Actually, they play a crucial role in the treatment of recursive calls, where the unfolding of recursion produces a tree-like structure whose size is just the number of times the (recursively defined) function will be used *globally*. Note that the value of a forest cardinality could also be undefined. For instance, this happens when infinite trees, corresponding to diverging recursive computations, are considered.

The expression $[\![\mathrm{I}]\!]_\rho^{\mathcal{E}}$ denotes the meaning of I, defined by induction along the lines of the previous discussion, where $\rho : \mathcal{V} \to \mathbb{N}$ is an assignment and $\mathcal{E}$ is an equational program giving meaning to the function symbols in I. Since $\mathcal{E}$ does not necessarily interpret such symbols as *total* functions, and moreover, the value of a forest cardinality can be undefined, $[\![\mathrm{I}]\!]_\rho^{\mathcal{E}}$ can be undefined itself. A *constraint* is an inequality in the form $\mathrm{I} \leqslant \mathrm{J}$. A constraint is *true* in an assignment ρ if $[\![\mathrm{I}]\!]_\rho^{\mathcal{E}}$ and $[\![\mathrm{J}]\!]_\rho^{\mathcal{E}}$ are both defined and the first is smaller or equal to the latter. Now, for a subset ϕ of $\mathcal{V}$, and for a set Φ of constraints involving variables in ϕ, the expression

$$\phi; \Phi \models^{\mathcal{E}} \mathrm{I} \leqslant \mathrm{J} \tag{3.3}$$

denotes the fact that the truth of $\mathrm{I} \leqslant \mathrm{J}$ semantically follows from the truth of the constraints in Φ. The expression $\phi; \Phi \models^{\mathcal{E}} \mathrm{I} \geqslant \mathbf{0}$ indicates that (the semantics of) I is *defined* for the relevant values of the variables in ϕ; this is usually written as $\phi; \Phi \models^{\mathcal{E}} \mathrm{I} \Downarrow$.

Similarly, one can define the meaning of expressions like $\phi; \Phi \models^{\mathcal{E}} \mathrm{I} = \mathrm{J}$ or $\phi; \Phi \models^{\mathcal{E}} \mathrm{I} \simeq \mathrm{J}$, the latter standing for the equality of I and J in the sense of Kleene, i.e. $\phi; \Phi \models^{\mathcal{E}} \mathrm{I} \Downarrow$ if and only if $\phi; \Phi \models^{\mathcal{E}} \mathrm{J} \Downarrow$, and if $\phi; \Phi \models^{\mathcal{E}} \mathrm{I} \Downarrow$ then $\phi; \Phi \models^{\mathcal{E}} \mathrm{I} = \mathrm{J}$. When both ϕ and Φ are empty, such expressions can be written in a much more concise form, e.g. $\mathrm{I} \simeq \mathrm{J}$ stands for $\varnothing; \varnothing \models^{\mathcal{E}} \mathrm{I} \simeq \mathrm{J}$.

The following two lemmas about forest cardinalities are useful, and will be crucial when proving the Substitution Lemma.

Lemma 3.1. *For every index terms* $\mathrm{I}, \mathrm{J}, \mathrm{K}, \mathrm{H}$*, we have:*

$$\bigtriangleup_a^{\mathrm{I}+\mathrm{J},\mathrm{K}} \mathrm{H} \simeq \bigtriangleup_a^{\mathrm{J},\mathrm{K}} \mathrm{H}\{a + \mathrm{I}/a\}.$$

Proof. The proof is by coinduction on the definition of $\bigtriangleup_a^{\mathrm{I}+\mathrm{J},\mathrm{K}} \mathrm{H}$ by distinguishing the cases for the different values of K. For $\mathrm{K} \simeq \mathbf{0}$ we have both:

$$\bigtriangleup_a^{\mathrm{I}+\mathrm{J},\mathbf{0}} \mathrm{H} \simeq \mathbf{0}; \qquad \bigtriangleup_a^{\mathrm{J},\mathbf{0}} \mathrm{H}\{a + \mathrm{I}/a\} \simeq \mathbf{0}.$$

For $\mathrm{K} \simeq \mathrm{L} + \mathbf{1}$ we have:

$$\bigtriangleup_a^{\mathrm{I}+\mathrm{J},\mathrm{L}+\mathbf{1}} \mathrm{H} \simeq \bigtriangleup_a^{\mathrm{I}+\mathrm{J},\mathrm{L}} \mathrm{H} + \mathbf{1} + \bigtriangleup_a^{\mathrm{I}+\mathrm{J}+\mathbf{1}+\bigtriangleup_a^{\mathrm{I}+\mathrm{J},\mathrm{L}} \mathrm{H}, \mathrm{H}\{\mathrm{I}+\mathrm{J}+\bigtriangleup_a^{\mathrm{I}+\mathrm{J},\mathrm{L}} \mathrm{H}/a\}} \mathrm{H},$$

and analogously

$$\bigtriangleup_a^{\mathrm{J},\mathrm{L}+\mathbf{1}} \mathrm{H}\{a + \mathrm{I}/a\} \simeq \bigtriangleup_a^{\mathrm{J},\mathrm{L}} \mathrm{H}\{a + \mathrm{I}/a\} + \mathbf{1} + \bigtriangleup_a^{\mathrm{J}+1+\bigtriangleup_a^{\mathrm{J},\mathrm{L}} \mathrm{H}\{a+\mathrm{I}/a\}, \mathrm{H}\{\mathrm{I}+\mathrm{J}+\bigtriangleup_a^{\mathrm{J},\mathrm{L}} \mathrm{H}\{a+\mathrm{I}/a\}/a\}} \mathrm{H}\{a + \mathrm{I}/a\}.$$

This concludes the proof. □

Lemma 3.2. *For every index term of the shape* $\bigcirc_a^{\boldsymbol{1},\mathrm{J}}\,\mathrm{I}$ *we have:*

$$\bigcirc_a^{\boldsymbol{1},\mathrm{J}}\,\mathrm{I} \simeq \sum_{b<\mathrm{J}} \bigcirc_a^{\boldsymbol{0},\boldsymbol{1}}\,\mathrm{I}\{a+\boldsymbol{1}+\bigcirc_a^{\boldsymbol{1},b}\,\mathrm{I}/a\}.$$

Proof. The proof is by coinduction on the definition of $\bigcirc_a^{\mathbf{1},\mathrm{J}}\,\mathrm{I}$ by distinguishing the cases for the different values of J. For $\mathrm{J} \simeq \mathbf{0}$, we have both:

$$\bigcirc_a^{\mathbf{1},\mathbf{0}}\,\mathrm{I} \simeq \mathbf{0}; \qquad \sum_{b<\mathbf{0}} \bigcirc_a^{\mathbf{0},\mathbf{1}}\,\mathrm{I}\{a+\mathbf{1}+\bigcirc_a^{\mathbf{1},b}\,\mathrm{I}/a\} \simeq \mathbf{0}.$$

For $\mathrm{J} \simeq \mathrm{L}+\mathbf{1}$ we have

$$\bigcirc_a^{\mathbf{1},\mathrm{L}+\mathbf{1}}\,\mathrm{I} \simeq \mathrm{K}+\mathbf{1}+\bigcirc_a^{\mathrm{K}+\mathbf{2},\mathrm{I}\{\mathrm{K}+\mathbf{1}/a\}}\,\mathrm{I}$$

and

$$\sum_{b<\mathrm{L}+\mathbf{1}} \bigcirc_a^{\mathbf{0},\mathbf{1}}\,\mathrm{I}\{a+\mathbf{1}+\bigcirc_a^{\mathbf{1},b}\,\mathrm{I}/a\} \simeq \mathrm{H}+\bigcirc_a^{\mathbf{0},\mathbf{1}}\,\mathrm{I}\{a+\mathbf{1}+\bigcirc_a^{\mathbf{1},\mathrm{L}}\,\mathrm{I}/a\},$$

where K is $\bigcirc_a^{\mathbf{1},\mathrm{L}}\,\mathrm{I}$ and H is $\sum_{b<\mathrm{L}} \bigcirc_a^{\mathbf{0},\mathbf{1}}\,\mathrm{I}\{a+\mathbf{1}+\bigcirc_a^{\mathbf{1},b}\,\mathrm{I}/a\}$. Now, by definition and by Lemma 3.1, we have

$$\bigcirc_a^{\mathbf{0},\mathbf{1}}\,\mathrm{I}\{a+\mathbf{1}+\bigcirc_a^{\mathbf{1},\mathrm{L}}\,\mathrm{I}/a\} \simeq \mathbf{1}+\bigcirc_a^{\mathbf{1},\mathrm{I}\{\mathrm{K}+\mathbf{1}/a\}}\,\mathrm{I}\{a+\mathbf{1}+\mathrm{K}/a\} \simeq \mathbf{1}+\bigcirc_a^{\mathrm{K}+\mathbf{2},\mathrm{I}\{\mathrm{K}+\mathbf{1}/a\}}\,\mathrm{I}.$$

This concludes the proof. □

Before embarking in the description of the type system, a further remark on the role of index terms could be useful. Index terms are not meant to be part of *programs* but of *types*. As a consequence, computation will not be carried out on index terms but on proper terms, which are the subject of Section 3.2 below.

3.2. **The Type System.** Terms are generated by the following grammar:

$$t ::= x \mid \underline{\mathtt{n}} \mid \mathtt{s}(t) \mid \mathtt{p}(t) \mid \lambda x.t \mid tu \mid \mathtt{ifz}\ t\ \mathtt{then}\ u\ \mathtt{else}\ v \mid \mathtt{fix}\ x.t$$

where $\mathtt{n}$ ranges over natural numbers and x ranges over a set of *variables*. As usual, terms which are equal modulo α-conversion are considered equal. This, in turn, allows to define the notion of substitution in the standard way. The set of *head subterms* of any term t can be defined easily by induction on the structure of t, e.g. the head subterms of $t = uv$ are t itself and the head subterms of u (but not those of v).

A notion of *size* $|t|$ for a term t will be useful in the sequel. This can be defined as follows:

$$|x| = 1; \qquad |\lambda x.t| = |t| + 1;$$
$$|\underline{\mathtt{n}}| = 1; \qquad |tu| = |t| + |u| + 1;$$
$$|\mathtt{s}(t)| = |t| + 2; \qquad |\ \mathtt{ifz}\ t\ \mathtt{then}\ u\ \mathtt{else}\ v| = |t| + |u| + |v| + 1;$$
$$|\mathtt{p}(t)| = |t| + 2; \qquad |\mathtt{fix}\ x.t| = |t| + 1.$$

$$\frac{}{\Gamma, x:\sigma \vdash x:\sigma} \qquad \frac{\Gamma, x:\sigma \vdash t:\tau}{\Gamma \vdash \lambda x.t : \sigma \to \tau} \qquad \frac{\Gamma \vdash t : \sigma \to \tau \quad \Gamma \vdash u : \sigma}{\Gamma \vdash tu : \tau}$$

$$\frac{}{\Gamma \vdash \underline{\mathtt{n}} : \mathtt{Nat}} \qquad \frac{\Gamma \vdash t : \mathtt{Nat}}{\Gamma \vdash \mathtt{s}(t) : \mathtt{Nat}} \qquad \frac{\Gamma \vdash t : \mathtt{Nat}}{\Gamma \vdash \mathtt{p}(t) : \mathtt{Nat}}$$

$$\frac{\Gamma \vdash t : \mathtt{Nat} \quad \Gamma \vdash u : \sigma \quad \Gamma \vdash v : \sigma}{\Gamma \vdash \ \mathtt{ifz}\ t\ \mathtt{then}\ u\ \mathtt{else}\ v : \sigma} \qquad \frac{\Gamma, x:\sigma \vdash t : \sigma}{\Gamma \vdash \mathtt{fix}\ x.t : \sigma}$$

Figure 2: The PCF Type System.

$$\frac{}{(\lambda x.t)u \to t\{u/x\}} \qquad \frac{}{\mathtt{s}(\underline{\mathtt{n}}) \to \underline{\mathtt{n}+\mathtt{1}}} \qquad \frac{}{\mathtt{p}(\underline{\mathtt{n}+\mathtt{1}}) \to \underline{\mathtt{n}}} \qquad \frac{}{\mathtt{p}(\underline{\mathtt{0}}) \to \underline{\mathtt{0}}}$$

$$\frac{}{\mathtt{ifz}\ \underline{\mathtt{0}}\ \mathtt{then}\ u\ \mathtt{else}\ v \to u} \qquad \frac{}{\mathtt{ifz}\ \underline{\mathtt{n}+\mathtt{1}}\ \mathtt{then}\ u\ \mathtt{else}\ v \to v}$$

$$\frac{}{\mathtt{fix}\ x.t \to t\{\mathtt{fix}\ x.t/x\}} \qquad \frac{t \to u}{\mathtt{s}(t) \to \mathtt{s}(u)} \qquad \frac{t \to u}{\mathtt{p}(t) \to \mathtt{p}(u)} \qquad \frac{t \to v}{tu \to vu}$$

$$\frac{t \to w}{\mathtt{ifz}\ t\ \mathtt{then}\ u\ \mathtt{else}\ v \to\ \mathtt{ifz}\ w\ \mathtt{then}\ u\ \mathtt{else}\ v}$$

Figure 3: Weak-head Reduction

Notice that for technical reasons size is defined in a slightly nonstandard way: every integer constant has size 1.

Lemma 3.3. *If t is a term and u is a subterm of t, then $|u| \leqslant |t|$.*

Terms can be typed by a well-known type system called PCF. Types are those generated by the basic type Nat and the binary type constructor $\to$. Typing rules are in Figure 2. A notion of weak-head reduction $\to$ can be easily defined: see Figure 3. A term t is said to be a *program* if it can be given the PCF type Nat in the empty context.

Almost all the definitions about dℓPCF in this and the next sections should be understood as parametric on an equational program $\mathcal{E}$ over a signature Σ. For the sake of simplicity, however, we will often avoid to explicitly mention $\mathcal{E}$ and leave it implicit.

dℓPCF can be seen as a refinement of PCF obtained by a linear decoration of its type derivations. Basic and modal types are defined as follows:

$$\sigma, \tau ::= \mathtt{Nat}[\mathrm{I}, \mathrm{J}] \ \mid\ A \multimap \sigma; \qquad \text{basic types}$$

$$A, B ::= [a < \mathrm{I}] \cdot \sigma; \qquad \text{modal types}$$

where I, J range over index terms and a ranges over index variables. $\mathtt{Nat}[\mathrm{I}]$ is syntactic sugar for $\mathtt{Nat}[\mathrm{I}, \mathrm{I}]$. Modal types need some comments. As a first approximation, they can be thought of as quantifiers over type variables. So, a type like $A = [a < \mathrm{I}] \cdot \sigma$ acts as a binder for the index variable a in the basic type σ. Moreover, the condition $a < \mathrm{I}$ says that A consists of all the instances of the basic type σ where the variable a is successively instantiated with the values from $\mathbf{0}$ to (the value of) $\mathrm{I} \dot{-} \mathbf{1}$, i.e. $\sigma\{\mathbf{0}/a\}, \ldots, \sigma\{\mathrm{I} \dot{-} \mathbf{1}/a\}$. For those readers who are familiar with linear logic, and in particular with BLL, the modal type $[a < \mathrm{I}] \cdot \sigma$ is a generalization of the BLL formula $!_{a<p}\sigma$ to arbitrary index terms. As such it can be thought of as representing the type $\sigma\{\mathbf{0}/a\} \otimes \cdots \otimes \sigma\{\mathrm{I} \dot{-} \mathbf{1}/a\}$. In analogy to what happens in the standard linear logic decomposition of the intuitionistic arrow,

$$\frac{\phi;\Phi \models^{\mathcal{E}} \mathrm{I} \Downarrow \quad \phi;\Phi \models^{\mathcal{E}} \mathrm{J} \Downarrow}{\phi;\Phi \vdash^{\mathcal{E}} \mathtt{Nat}[\mathrm{I},\mathrm{J}] \Downarrow}\ (\mathtt{Nat}.t) \qquad \frac{\phi;\Phi \vdash^{\mathcal{E}} A \Downarrow \quad \phi;\Phi \vdash^{\mathcal{E}} \sigma \Downarrow}{\phi;\Phi \vdash^{\mathcal{E}} A \multimap \sigma \Downarrow}\ (\multimap .t)$$

$$\frac{\phi, a;\Phi, a < \mathrm{I} \vdash^{\mathcal{E}} \sigma \Downarrow \quad \phi;\Phi \models^{\mathcal{E}} \mathrm{I} \Downarrow}{\phi;\Phi \vdash^{\mathcal{E}} [a < \mathrm{I}] \cdot \sigma \Downarrow}\ ([-] \cdot .t)$$

Figure 4: Well-defined Types

$$\frac{\phi;\Phi \models^{\mathcal{E}} \mathrm{K} \leqslant \mathrm{I} \quad \phi;\Phi \models^{\mathcal{E}} \mathrm{J} \leqslant \mathrm{H}}{\phi;\Phi \vdash^{\mathcal{E}} \mathtt{Nat}[\mathrm{I},\mathrm{J}] \sqsubseteq \mathtt{Nat}[\mathrm{K},\mathrm{H}]}\ (\mathtt{Nat}.l) \qquad \frac{\phi;\Phi \vdash^{\mathcal{E}} B \sqsubseteq A \quad \phi;\Phi \vdash^{\mathcal{E}} \sigma \sqsubseteq \tau}{\phi;\Phi \vdash^{\mathcal{E}} A \multimap \sigma \sqsubseteq B \multimap \tau}\ (\multimap .l)$$

$$\frac{\phi, a;\Phi, a < \mathrm{I} \vdash^{\mathcal{E}} \sigma \sqsubseteq \tau \quad \phi;\Phi \models^{\mathcal{E}} \mathrm{J} \leqslant \mathrm{I}}{\phi;\Phi \vdash^{\mathcal{E}} [a < \mathrm{I}] \cdot \sigma \sqsubseteq [a < \mathrm{J}] \cdot \tau}\ ([-] \cdot .l)$$

Figure 5: The Subtyping Relation

i.e. $!A \multimap B = A \Rightarrow B$, it is sufficient to restrict to modal types appearing in negative position.Finally, for those readers with some knowledge of DML, modal types are in a way similar to DML's subset sort constructions [35].

We always assume that index terms appearing inside types are defined for all the relevant values of the variables in ϕ. This is captured by the judgement $\phi;\Phi \vdash^{\mathcal{E}} \sigma \Downarrow$, whose rules are in Figure 4.

In the typing rules, modal types need to be manipulated in an algebraic way. For this reason, two operations on modal types need to be introduced. The first one is a binary operation $\uplus$ on modal types. Suppose that $A = [a < \mathrm{I}] \cdot \mu\{a/c\}$ and that $B = [b < \mathrm{J}] \cdot \mu\{\mathrm{I} + b/c\}$. In other words, A consists of the first I instances of μ, i.e. $\mu\{\mathbf{0}/c\}, \ldots, \mu\{\mathrm{I} \dot{-} \mathbf{1}/c\}$ while B consists of the next J instances of μ, i.e. $\mu\{\mathrm{I} + \mathbf{0}/c\}, \ldots, \mu\{\mathrm{I} + \mathrm{J} \dot{-} \mathbf{1}/c\}$. Their *sum* $A \uplus B$ is naturally defined as a modal type consisting of the first $\mathrm{I} + \mathrm{J}$ instances of μ, i.e. $[c < \mathrm{I} + \mathrm{J}] \cdot \mu$. An operation of bounded sum on modal types can be defined by generalizing the idea above. Suppose that $A = [b < \mathrm{J}] \cdot \sigma\{\sum_{d<a} \mathrm{J}\{d/a\} + b/c\}$. Then its *bounded sum* $\sum_{a<\mathrm{I}} A$ is $[c < \sum_{a<\mathrm{I}} \mathrm{J}] \cdot \sigma$.

To every type σ corresponds a type $(\!|\sigma|\!)$ of ordinary PCF, namely a type built from the basic type $\mathtt{Nat}$ and the arrow operator $\rightarrow$:

$$(\!|\mathtt{Nat}[\mathrm{I},\mathrm{J}]|\!) = \mathtt{Nat};$$
$$(\!|[a < \mathrm{I}] \cdot \sigma \multimap \tau|\!) = (\!|\sigma|\!) \rightarrow (\!|\tau|\!).$$

Central to $\mathsf{d}\ell\mathsf{PCF}$ is the notion of subtyping. An inequality relation $\sqsubseteq$ between (basic and modal) types can be defined by way of the formal system in Figure 5. This relation corresponds to lifting index inequalities at the type level. The equivalence $\phi;\Phi \vdash \sigma \cong \tau$

holds when both $\phi; \Phi \vdash \sigma \sqsubseteq \tau$ and $\phi; \Phi \vdash \tau \sqsubseteq \sigma$ can be derived from the rules in Figure 5. $\phi; \Phi \vdash \sigma \Downarrow$ is syntactic sugar for $\phi; \Phi \vdash \sigma \sqsubseteq \sigma$.

It is now time to introduce the main object of this paper, namely the type system $\mathsf{d}\ell\mathsf{PCF}$. *Typing judgements* of $\mathsf{d}\ell\mathsf{PCF}$ are expressions in the form

$$\phi; \Phi; \Gamma \vdash_{\mathrm{I}}^{\mathcal{E}} t : \sigma, \tag{3.4}$$

where Γ is a *typing context*, that is, a set of term variable assignments of the shape $x : A$ where each variable x occurs at most once. The expression (3.4) can be informally read as follows: for every values of the index variables in ϕ satisfying Φ, t can be given type σ and *cost* I once its free term variables have types as in Γ. In proving this, equations from $\mathcal{E}$ can play a role.

Typing rules are in Figure 6, where binary and bounded sums are used in their natural generalization to contexts. A *type derivation* is nothing more than a tree built according to typing rules. A *precise type derivation* is a type derivation such that all premises in the form $\sigma \sqsubseteq \tau$ (respectively, in the form $\mathrm{I} \leqslant \mathrm{J}$) are required to be in the form $\sigma \cong \tau$ (respectively, $\mathrm{I} = \mathrm{J}$).

First of all, observe that the typing rules are syntax-directed: given a term t, all type derivations for t end with the same typing rule, namely the one corresponding to the last syntax rule used in building t. In particular, no explicit subtyping rule exists, but subtyping is applied to the context in every rule. A syntax-directed type system offers a key advantage: it allows one to prove the statements about type derivations by induction on the structure of terms. This greatly simplifies the proof of crucial properties like subject reduction.

Typing rules have premises of three different kinds:

- Of course, typing a term requires typing its immediate subterms, so typing judgements can be rule premises.
- As just mentioned, typing rules allow to subtype the context Γ, so subtyping judgements can be themselves rule premises.
- The application of typing rules (and also of subtyping rules, see Figure 5) sometimes depends on the truth of some inequalities between index terms in the model induced by $\mathcal{E}$.

As a consequence, typing rules can only be applied if some relations between index terms are consequences of the constraints in Φ. These assumptions have a semantic nature, but could of course be verified by any sound formal system. Completeness (see Section 5), however, only holds if all *true* inequalities can be used as assumptions. As a consequence, type inference but also type (derivation) checking are bound to be problematic from a computational point of view. See Section 6 for a more thorough discussion on this issue.

As a last remark, note that each rule can be seen as a *decoration* of a rule of ordinary PCF. More: for every $\mathsf{d}\ell\mathsf{PCF}$ type derivation π of $\phi; \Phi; \Gamma \vdash_{\mathrm{I}}^{\mathcal{E}} t : \sigma$ there is a structurally identical derivation in PCF for the same term, i.e. a derivation $(\!|\pi|\!) \rhd (\!|\Gamma|\!) \vdash t : (\!|\sigma|\!)$.

3.3. **An Example.** In this section, we will show how $\mathsf{d}\ell\mathsf{PCF}$ can give a sensible type to the example we talked about in the Introduction, namely

$$\mathtt{dbl} = \mathtt{fix}\ f.\lambda x.\ \mathtt{ifz}\ x\ \mathtt{then}\ \underline{0}\ \mathtt{else}\ \mathtt{s}(\mathtt{s}(f(\mathtt{p}(x)))).$$

First, let us take a look at a subterm of $\mathtt{dbl}$, namely $t = \ \mathtt{ifz}\ x\ \mathtt{then}\ \underline{0}\ \mathtt{else}\ \mathtt{s}(\mathtt{s}(f(\mathtt{p}(x))))$. In plain PCF, t receives the type $\mathtt{Nat}$ in an environment where x has type $\mathtt{Nat}$ and f has type $\mathtt{Nat} \to \mathtt{Nat}$. Presumably, a $\mathsf{d}\ell\mathsf{PCF}$ type for t can be obtained by decorating in an

$$\frac{\begin{array}{c}\phi;\Phi \models^{\mathcal{E}} \mathbf{0} \leqslant \mathrm{J} \qquad \phi;\Phi \models^{\mathcal{E}} \mathbf{1} \leqslant \mathrm{I} \\ \phi;\Phi \vdash^{\mathcal{E}} \sigma\{\mathbf{0}/a\} \sqsubseteq \tau \\ \phi;\Phi \vdash^{\mathcal{E}} ([a < \mathrm{I}] \cdot \sigma) \Downarrow \qquad \phi;\Phi \vdash^{\mathcal{E}} \Gamma \Downarrow\end{array}}{\phi;\Phi;\Gamma, x : [a < \mathrm{I}] \cdot \sigma \vdash^{\mathcal{E}}_{\mathrm{J}} x : \tau}\ V \qquad \frac{\phi;\Phi;\Gamma, x : [a < \mathrm{I}] \cdot \sigma \vdash^{\mathcal{E}}_{\mathrm{J}} t : \tau}{\phi;\Phi;\Gamma \vdash^{\mathcal{E}}_{\mathrm{J}} \lambda x.t : [a < \mathrm{I}] \cdot \sigma \multimap \tau}\ L$$

$$\frac{\begin{array}{c}\phi;\Phi;\Gamma \vdash^{\mathcal{E}}_{\mathrm{J}} t : [a < \mathrm{I}] \cdot \sigma \multimap \tau \\ \phi, a;\Phi, a < \mathrm{I};\Delta \vdash^{\mathcal{E}}_{\mathrm{K}} u : \sigma \\ \phi;\Phi \vdash^{\mathcal{E}} \Sigma \sqsubseteq \Gamma \uplus \sum_{a<\mathrm{I}} \Delta \\ \phi;\Phi \models^{\mathcal{E}} \mathrm{H} \geqslant \mathrm{J} + \mathrm{I} + \sum_{a<\mathrm{I}} \mathrm{K}\end{array}}{\phi;\Phi;\Sigma \vdash^{\mathcal{E}}_{\mathrm{H}} tu : \tau}\ A \qquad \frac{\begin{array}{c}\phi;\Phi \vdash^{\mathcal{E}} \mathtt{Nat}[\mathrm{I} + \mathbf{1}, \mathrm{J} + \mathbf{1}] \sqsubseteq \mathtt{Nat}[\mathrm{K}, \mathrm{H}] \\ \phi;\Phi;\Gamma \vdash^{\mathcal{E}}_{\mathrm{L}} t : \mathtt{Nat}[\mathrm{I}, \mathrm{J}]\end{array}}{\phi;\Phi;\Gamma \vdash^{\mathcal{E}}_{\mathrm{L}} \mathtt{s}(t) : \mathtt{Nat}[\mathrm{K}, \mathrm{H}]}\ S$$

$$\frac{\begin{array}{c}\phi;\Phi \models^{\mathcal{E}} \mathrm{K} \geqslant \mathbf{0} \\ \phi;\Phi \models^{\mathcal{E}} \mathrm{I} \leqslant \mathbf{n} \\ \phi;\Phi \models^{\mathcal{E}} \mathbf{n} \leqslant \mathrm{J} \\ \phi;\Phi \vdash^{\mathcal{E}} \Gamma \Downarrow\end{array}}{\phi;\Phi;\Gamma \vdash^{\mathcal{E}}_{\mathrm{K}} \underline{\mathtt{n}} : \mathtt{Nat}[\mathrm{I}, \mathrm{J}]}\ N \qquad \frac{\begin{array}{c}\phi;\Phi \vdash^{\mathcal{E}} \mathtt{Nat}[\mathrm{I} \dot{-} \mathbf{1}, \mathrm{J} \dot{-} \mathbf{1}] \sqsubseteq \mathtt{Nat}[\mathrm{K}, \mathrm{H}] \\ \phi;\Phi;\Gamma \vdash^{\mathcal{E}}_{\mathrm{L}} t : \mathtt{Nat}[\mathrm{I}, \mathrm{J}]\end{array}}{\phi;\Phi;\Gamma \vdash^{\mathcal{E}}_{\mathrm{L}} \mathtt{p}(t) : \mathtt{Nat}[\mathrm{K}, \mathrm{H}]}\ P$$

$$\frac{\begin{array}{c}\phi;\Phi;\Gamma \vdash^{\mathcal{E}}_{\mathrm{K}} t : \mathtt{Nat}[\mathrm{I}, \mathrm{J}] \\ \phi;\Phi, \mathrm{I} \leqslant \mathtt{0};\Delta \vdash^{\mathcal{E}}_{\mathrm{H}} u : \sigma \\ \phi;\Phi, \mathrm{J} \geqslant \mathbf{1};\Delta \vdash^{\mathcal{E}}_{\mathrm{H}} v : \sigma \\ \phi;\Phi \vdash^{\mathcal{E}} \Sigma \sqsubseteq \Gamma \uplus \Delta \\ \phi;\Phi \models^{\mathcal{E}} \mathrm{L} \geqslant \mathrm{K} + \mathrm{H}\end{array}}{\phi;\Phi;\Sigma \vdash^{\mathcal{E}}_{\mathrm{L}} \ \mathtt{ifz}\ t\ \mathtt{then}\ u\ \mathtt{else}\ v : \sigma}\ F$$

$$\frac{\begin{array}{c}\phi, b;\Phi, b < \mathrm{L};\Gamma, x : [a < \mathrm{I}] \cdot \sigma \vdash^{\mathcal{E}}_{\mathrm{K}} t : \tau \\ \phi;\Phi \vdash^{\mathcal{E}} \tau\{\mathbf{0}/b\} \sqsubseteq \mu \\ \phi, a, b;\Phi, a < \mathrm{I}, b < \mathrm{L} \vdash^{\mathcal{E}} \tau\{\triangle_b^{b+\mathbf{1},a} \mathrm{I} + b + \mathbf{1}/b\} \sqsubseteq \sigma \\ \phi;\Phi \vdash^{\mathcal{E}} \Sigma \sqsubseteq \sum_{b<\mathrm{L}} \Gamma \\ \phi;\Phi \models^{\mathcal{E}} \triangle_b^{\mathbf{0},\mathbf{1}} \mathrm{I} \leqslant \mathrm{L}, \mathrm{M} \\ \phi;\Phi \models^{\mathcal{E}} \mathrm{N} \geqslant \mathrm{M} \dot{-} \mathbf{1} + \sum_{b<\mathrm{L}} \mathrm{K}\end{array}}{\phi;\Phi;\Sigma \vdash^{\mathcal{E}}_{\mathrm{N}} \mathtt{fix}\ x.t : \mu}\ R$$

Figure 6: Typing Rules

appropriate way the type above. In other words, we are looking for a type derivation with conclusion:

$$\phi;\Phi;x : [a < \mathrm{I}] \cdot \mathtt{Nat}[\mathrm{J}], f : [b < \mathrm{K}] \cdot ([c < \mathrm{H}] \cdot \mathtt{Nat}[\mathrm{L}] \multimap \mathtt{Nat}[\mathrm{M}]) \vdash^{\mathcal{E}}_{\mathrm{N}} t : \mathtt{Nat}[\mathrm{P}].$$

But how should we proceed? What we would like, at the end of the day, is being able to describe how the value of t depends on the value of x, so we could look for a type derivation in this form:

$$d;\varnothing;x : [\mathrm{I}] \cdot \mathtt{Nat}[d], f : [b < \mathrm{K}] \cdot ([\mathrm{H}] \cdot \mathtt{Nat}[d \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(d \dot{-} \mathbf{1})]) \vdash^{\mathcal{E}}_{\mathrm{N}} t : \mathtt{Nat}[\mathbf{2}d],$$

where $[a < \mathrm{I}]$ (respectively, $[c < \mathrm{H}]$) has been abbreviated into [I] (respectively, [H]) because the bound variable a (respectively, c) does not appear free in the underlying type. But how to give values to I, K, and H? One could be tempted to define I simply as $\mathbf{2}$, since there are two occurrences of x in t. However, in view of the role played by x and f in `dbl`, I should be rather defined taking into account the number of times x will be copied along the computation of `dbl` on *any* input. A good guess could be, for example, $d + \mathbf{1}$. Similarly, H could be d. But how about K? How many times f is used? If $d = 0$, then f is not called, while if $d > \mathbf{0}$, the function is called once. In other words, a guess for H could be $\mathtt{gt}(d, \mathbf{0})$. Here we use the infix notation $>$ for the operator `gt` just to improve readability. Let us now try to build a derivation for

$$d; \varnothing; x : [d + \mathbf{1}] \cdot \mathtt{Nat}[d], f : [d > \mathbf{0}] \cdot ([d] \cdot \mathtt{Nat}[d \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(d \dot{-} \mathbf{1})]) \vdash^{\mathcal{E}}_{\mathbf{0}} t : \mathtt{Nat}[\mathbf{2}d].$$

Actually, it has the following shape:

$$\frac{\begin{array}{c}\pi \rhd d; \varnothing; x : [\mathbf{1}] \cdot \mathtt{Nat}[d] \vdash^{\mathcal{E}}_{\mathbf{0}} x : \mathtt{Nat}[d] \\ \rho \rhd d; d \leqslant \mathbf{0}; x : [d] \cdot \mathtt{Nat}[d], f : [d > \mathbf{0}] \cdot ([d] \cdot \mathtt{Nat}[d \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(d \dot{-} \mathbf{1})]) \vdash^{\mathcal{E}}_{\mathbf{0}} \underline{\mathbf{0}} : \mathtt{Nat}[\mathbf{2}d] \\ \nu \rhd d; d > \mathbf{0}; x : [d] \cdot \mathtt{Nat}[d], f : [d > \mathbf{0}] \cdot ([d] \cdot \mathtt{Nat}[d \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(d \dot{-} \mathbf{1})]) \vdash^{\mathcal{E}}_{\mathbf{0}} \mathtt{s}(\mathtt{s}(f(\mathtt{p}(x)))) : \mathtt{Nat}[\mathbf{2}d]\end{array}}{d; \varnothing; x : [d + \mathbf{1}] \cdot \mathtt{Nat}[d], f : [d > \mathbf{0}] \cdot ([d] \cdot \mathtt{Nat}[d \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(d \dot{-} \mathbf{1})]) \vdash^{\mathcal{E}}_{\mathbf{0}} t : \mathtt{Nat}[\mathbf{2}d]}$$

where assignments to types in the form $[\mathbf{0}] \cdot \sigma$ have been omitted from contexts. Now, π and ρ can be easily built, while ν requires a little effort: it is the type derivation

$$\cfrac{\cfrac{\cfrac{\begin{array}{c}\mu \rhd d; d > \mathbf{0}; f : [d > \mathbf{0}] \cdot ([d] \cdot \mathtt{Nat}[d \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(d \dot{-} \mathbf{1})]) \vdash^{\mathcal{E}}_{\mathbf{0}} f : [d] \cdot \mathtt{Nat}[d \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(d \dot{-} \mathbf{1})] \\ \xi \rhd d; d > \mathbf{0}; x : [\mathbf{1}] \cdot \mathtt{Nat}[d] \vdash^{\mathcal{E}}_{\mathbf{0}} \mathtt{p}(x) : \mathtt{Nat}[d \dot{-} \mathbf{1}]\end{array}}{d; d > \mathbf{0}; x : [d] \cdot \mathtt{Nat}[d], f : [d > \mathbf{0}] \cdot ([d] \cdot \mathtt{Nat}[d \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(d \dot{-} \mathbf{1})]) \vdash^{\mathcal{E}}_{\mathbf{0}} f(\mathtt{p}(x)) : \mathtt{Nat}[\mathbf{2}(d \dot{-} \mathbf{1})]}}{d; d > \mathbf{0}; x : [d] \cdot \mathtt{Nat}[d], f : [d > \mathbf{0}] \cdot ([d] \cdot \mathtt{Nat}[d \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(d \dot{-} \mathbf{1})]) \vdash^{\mathcal{E}}_{\mathbf{0}} \mathtt{s}(f(\mathtt{p}(x))) : \mathtt{Nat}[\mathbf{2}d \dot{-} \mathbf{1}]}}{d; d > \mathbf{0}; x : [d] \cdot \mathtt{Nat}[d], f : [d > \mathbf{0}] \cdot ([d] \cdot \mathtt{Nat}[d \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(d \dot{-} \mathbf{1})]) \vdash^{\mathcal{E}}_{\mathbf{0}} \mathtt{s}(\mathtt{s}(f(\mathtt{p}(x)))) : \mathtt{Nat}[\mathbf{2}d]}$$

where μ and ξ are themselves easily definable. Summing up, t can indeed be given the type we wanted it to have. As a consequence, we can say that

$$d; \varnothing; f : [d > \mathbf{0}] \cdot ([d] \cdot \mathtt{Nat}[d \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(d \dot{-} \mathbf{1})]) \vdash^{\mathcal{E}}_{\mathbf{0}} \lambda x.t : [d + \mathbf{1}] \cdot \mathtt{Nat}[d] \multimap \mathtt{Nat}[\mathbf{2}d].$$

However, we have only solved half of the problem, since the last step (namely typing the fixpoint) is definitely the most complicated. In particular, the rule R requires an index variable b which somehow ranges over all recursive calls. A different but related type can be given to $\lambda x.t$, namely

$$\begin{array}{c} a, b; b < a + \mathbf{1}; f : [a > b] \cdot ([a \dot{-} b] \cdot \mathtt{Nat}[a \dot{-} b \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(a \dot{-} b \dot{-} \mathbf{1})]) \\ \vdash^{\mathcal{E}}_{\mathbf{0}} \lambda x.t : [a \dot{-} b + \mathbf{1}] \cdot \mathtt{Nat}[a \dot{-} b] \multimap \mathtt{Nat}[\mathbf{2}(a \dot{-} b)]. \end{array}$$

By the way, this does not require rebuilding the entire type derivation (see the properties in the forthcoming Section 3.4). Let us now check whether the judgement above can be the premise of the rule R. Following the notation in the typing rule R we can stipulate that:

$$\begin{aligned} \mathrm{I} &\equiv a > b; \\ \mathrm{J} &\equiv a; \\ \mathrm{K} &\equiv \mathbf{0}; \\ \mathrm{L} &\equiv a + \mathbf{1}; \end{aligned}$$

and

$$\sigma \equiv [a \dot{-} b] \cdot \mathtt{Nat}[a \dot{-} b \dot{-} \mathbf{1}] \multimap \mathtt{Nat}[\mathbf{2}(a \dot{-} b \dot{-} \mathbf{1})];$$
$$\tau \equiv [a \dot{-} b + \mathbf{1}] \cdot \mathtt{Nat}[a \dot{-} b] \multimap \mathtt{Nat}[\mathbf{2}(a \dot{-} b)];$$
$$\mu \equiv \tau\{\mathbf{0}/b\} \equiv [a + \mathbf{1}] \cdot \mathtt{Nat}[a] \multimap \mathtt{Nat}[\mathbf{2}a];$$
$$\Gamma \equiv \Sigma \equiv \varnothing.$$

We can then conclude that, since $a < (a > b)$ implies $a = \mathbf{0}$:

$$a; \varnothing \models \bigcirc\!\!\!\!\!\!\!\!\!\!\begin{array}{c}\mathbf{0},\mathbf{1}\\ \triangle\\ b\end{array} \mathrm{I} = a + \mathbf{1} = \mathrm{J};$$

$$a, b; a < (a > b) \models \bigcirc\!\!\!\!\!\!\!\!\!\!\begin{array}{c}b+\mathbf{1},a\\ \triangle\\ b\end{array} \mathrm{I} = \mathbf{0};$$

$$a; \varnothing \models \tau\{ \bigcirc\!\!\!\!\!\!\!\!\!\!\begin{array}{c}b+\mathbf{1},a\\ \triangle\\ b\end{array} \mathrm{I} + b + \mathbf{1}/b\} = \tau\{b + \mathbf{1}/b\} = \sigma;$$

and, ultimately, that $a; \varnothing; \varnothing \vdash^{\mathcal{E}}_{a} \mathtt{dbl} : \mu$.

3.4. **Properties.** This section is mainly concerned with Subject Reduction. Subject Reduction will only be proved for closed terms, since the language is endowed with a weak notion of reduction and, as a consequence, reduction cannot happen in the scope of lambda abstractions. The system $\mathsf{d}\ell\mathsf{PCF}$ enjoys some nice properties that are both necessary intermediate steps towards proving subject reduction and essential ingredients for proving soundness and relative completeness. These properties permit to manipulate judgements being sure that derivability is preserved.

First of all, the constraints Φ in a typing judgement can be made stronger without altering the rest:

Lemma 3.4 (Constraint Strenghtening)**.** *Let $\phi; \Phi; \Gamma \vdash_{\mathrm{I}} t : \sigma$ and $\phi; \Psi \models^{\mathcal{E}} \Phi$. Then, $\phi; \Psi; \Gamma \vdash_{\mathrm{I}} t : \sigma$.*

Proof. It follows easily by definition of $\phi; \Psi \models^{\mathcal{E}} \Phi$. □

Note that a sort of strengthening also holds for weights.

Lemma 3.5 (Weight Monotonicity)**.** *Let $\phi; \Phi; \Gamma \vdash_{\mathrm{I}} t : \sigma$ and $\phi; \Phi \models^{\mathcal{E}} \mathrm{I} \leqslant \mathrm{J}$. Then, $\phi; \Phi; \Gamma \vdash_{\mathrm{J}} t : \sigma$.*

Proof. It follows easily by induction on the derivation proving $\phi; \Phi; \Gamma \vdash_{\mathrm{I}} t : \sigma$. In particular, observe that all rules altering the weight are designed in such a way as to allow the latter to be lifted up. □

Whenever a parameter in a subtyping judgment needs to be specialized, we can simply substitute it with an index term.

Lemma 3.6 (Index Term Substitution Respects Subtyping)**.** *Let $\phi, a; \Phi \vdash \theta \sqsubseteq \gamma$ and I be an index term. Then, $\phi; \Phi\{\mathrm{I}/a\}, \Psi \vdash \theta\{\mathrm{I}/a\} \sqsubseteq \gamma\{\mathrm{I}/a\}$ whenever $\phi; \Psi \models \mathrm{I} \Downarrow$.*

Proof. Easy. □

Subtyping can be freely applied both to the context Γ (contravariantly) and to the type σ (covariantly), leaving the rest of the judgement unchanged:

Lemma 3.7 (Subtyping)**.** *Suppose* $\phi; \Phi; x_1 : A_1, \ldots, x_n : A_n \vdash_{\mathrm{I}} t : \sigma$ *and* $\phi; \Phi \vdash B_i \sqsubseteq A_i$ *for* $1 \leqslant i \leqslant n$ *and* $\phi; \Phi \vdash \sigma \sqsubseteq \tau$*. Then,* $\phi; \Phi; x_1 : B_1, \ldots, x_n : B_n \vdash_{\mathrm{I}} t : \tau$*.*

Proof. By induction on the structure of a derivation π for

$$\phi; \Phi; x_1 : A_1, \ldots, x_n : A_n \vdash_{\mathrm{I}} t : \sigma.$$

Let us examine some interesting cases:

- If π is just
$$\dfrac{\begin{array}{c}\phi; \Phi \models^{\mathcal{E}} \mathbf{0} \leqslant \mathrm{J} \qquad \phi; \Phi \models^{\mathcal{E}} \mathbf{1} \leqslant \mathrm{I} \\ \phi; \Phi \vdash^{\mathcal{E}} \mu\{\mathbf{0}/a\} \sqsubseteq \sigma \\ \phi; \Phi \vdash^{\mathcal{E}} ([a < \mathrm{I}] \cdot \mu) \Downarrow \qquad \phi; \Phi \vdash^{\mathcal{E}} \Gamma \Downarrow\end{array}}{\phi; \Phi; \Gamma, x : [a < \mathrm{I}] \cdot \mu \vdash^{\mathcal{E}}_{\mathrm{J}} x : \sigma}\ V$$
then, by assumption we have that $B \equiv [a < \mathrm{K}] \cdot \gamma$ and $\phi; \Phi \vdash [a < \mathrm{K}] \cdot \gamma \sqsubseteq [a < \mathrm{I}] \cdot \mu$. Moreover, by assumption we have $\phi; \Phi \vdash \sigma \sqsubseteq \tau$. From $\phi; \Phi \vdash [a < \mathrm{K}] \cdot \gamma \sqsubseteq [a < \mathrm{I}] \cdot \mu$, it follows that $\phi; \Phi, a < \mathrm{K} \vdash \gamma \sqsubseteq \mu$ and that $\phi; \Phi \models \mathrm{I} \leqslant \mathrm{K}$. By Lemma 3.6, $\phi; \Phi \vdash \gamma\{\mathbf{0}/a\} \sqsubseteq \mu\{\mathbf{0}/a\}$, which by transitivity of $\sqsubseteq$ implies $\phi; \Phi \vdash^{\mathcal{E}} \gamma\{\mathbf{0}/a\} \sqsubseteq \tau$. Now, if Δ is a context such that (with a slight abuse of notation) $\phi; \Phi \vdash^{\mathcal{E}} \Delta \sqsubseteq \Gamma$, then $\phi; \Phi \vdash^{\mathcal{E}} \Delta \Downarrow$. Summing up,
$$\dfrac{\begin{array}{c}\phi; \Phi \models^{\mathcal{E}} \mathbf{0} \leqslant \mathrm{J} \qquad \phi; \Phi \models^{\mathcal{E}} \mathbf{1} \leqslant \mathrm{K} \\ \phi; \Phi \vdash^{\mathcal{E}} \gamma\{\mathbf{0}/a\} \sqsubseteq \tau \\ \phi; \Phi \vdash^{\mathcal{E}} ([a < \mathrm{K}] \cdot \gamma) \Downarrow \qquad \phi; \Phi \vdash^{\mathcal{E}} \Delta \Downarrow\end{array}}{\phi; \Phi; \Delta, x : [a < \mathrm{K}] \cdot \gamma \vdash^{\mathcal{E}}_{\mathrm{J}} x : \tau}\ V$$
- If π is
$$\dfrac{\begin{array}{c}\phi; \Phi; \Gamma \vdash^{\mathcal{E}}_{\mathrm{J}} t : [a < \mathrm{I}] \cdot \mu \multimap \sigma \\ \phi, a; \Phi, a < \mathrm{I}; \Delta \vdash^{\mathcal{E}}_{\mathrm{K}} u : \mu \\ \phi; \Phi \vdash^{\mathcal{E}} \Sigma \sqsubseteq \Gamma \uplus \sum_{a<\mathrm{I}} \Delta \\ \phi; \Phi \models^{\mathcal{E}} \mathrm{H} \geqslant \mathrm{J} + \mathrm{I} + \sum_{a<\mathrm{I}} \mathrm{K}\end{array}}{\phi; \Phi; \Sigma \vdash^{\mathcal{E}}_{\mathrm{H}} tu : \sigma}\ A$$
but we have $\phi; \Phi \vdash^{\mathcal{E}} \sigma \sqsubseteq \tau$ and $\phi; \Phi \vdash^{\mathcal{E}} \Theta \sqsubseteq \Sigma$, then by induction hypothesis we can easily conclude that $\phi; \Phi; \Gamma \vdash^{\mathcal{E}}_{\mathrm{J}} t : [a < \mathrm{I}] \cdot \mu \multimap \tau$ and, by transitivity of $\sqsubseteq$, that $\phi; \Phi \vdash^{\mathcal{E}} \Theta \sqsubseteq \Gamma \uplus \sum_{a<\mathrm{I}} \Delta$. As a consequence:
$$\dfrac{\begin{array}{c}\phi; \Phi; \Gamma \vdash^{\mathcal{E}}_{\mathrm{J}} t : [a < \mathrm{I}] \cdot \mu \multimap \tau \\ \phi, a; \Phi, a < \mathrm{I}; \Delta \vdash^{\mathcal{E}}_{\mathrm{K}} u : \mu \\ \phi; \Phi \vdash^{\mathcal{E}} \Theta \sqsubseteq \Gamma \uplus \sum_{a<\mathrm{I}} \Delta \\ \phi; \Phi \models^{\mathcal{E}} \mathrm{H} \geqslant \mathrm{J} + \mathrm{I} + \sum_{a<\mathrm{I}} \mathrm{K}\end{array}}{\phi; \Phi; \Theta \vdash^{\mathcal{E}}_{\mathrm{H}} tu : \tau}\ A$$

The other cases are similar. □

Weakening holds for term contexts:

Lemma 3.8 (Context Weakening)**.** *Let* $\phi; \Phi; \Gamma \vdash_{\mathrm{I}} t : \sigma$*. Then,* $\phi; \Phi; \Gamma, \Delta \vdash_{\mathrm{I}} t : \sigma$ *whenever* $\phi; \Phi \vdash \Delta \Downarrow$*.*

Proof. Easy, by induction on the derivation proving $\phi; \Phi; \Gamma \vdash_{\mathrm{I}} t : \sigma$. □

Another useful transformation on type derivations is substitution of an index variable for an index term:

Lemma 3.9 (Index Term Substitution)**.** *Let* $\phi, a; \Phi; \Gamma \vdash_{\mathrm{I}} t : \sigma$*. Then we have*

$$\phi; \Phi\{\mathrm{J}/a\}, \Psi; \Gamma\{\mathrm{J}/a\} \vdash_{\mathrm{I}\{\mathrm{J}/a\}} t : \sigma\{\mathrm{J}/a\}$$

for every J *such that* $\phi, \Psi \models^{\mathcal{E}} \mathrm{J} \Downarrow$*.*

Proof. By induction on the structure of a derivation π for

$$\phi, a; \Phi; \Gamma \vdash_{\mathrm{I}} t : \sigma.$$

Let us examine some cases:

- If π is just
$$\frac{\begin{array}{c}\phi, a; \Phi \models^{\mathcal{E}} \mathbf{0} \leqslant \mathrm{I} \qquad \phi, a; \Phi \models^{\mathcal{E}} \mathbf{1} \leqslant \mathrm{K} \\ \phi, a; \Phi \vdash^{\mathcal{E}} \mu\{\mathbf{0}/b\} \sqsubseteq \sigma \\ \phi, a; \Phi \vdash^{\mathcal{E}} ([b < \mathrm{K}] \cdot \mu) \Downarrow \qquad \phi, a; \Phi \vdash^{\mathcal{E}} \Gamma \Downarrow\end{array}}{\phi, a; \Phi; \Gamma, x : [b < \mathrm{K}] \cdot \mu \vdash^{\mathcal{E}}_{\mathrm{I}} x : \sigma}\ V$$
then of course we have that $\phi; \Phi\{\mathrm{J}/a\}, \Psi \models^{\mathcal{E}} \mathbf{0} \leqslant \mathrm{I}\{\mathrm{J}/a\}$ and that $\phi; \Phi\{\mathrm{J}/a\}, \Psi \models^{\mathcal{E}} \mathbf{1} \leqslant \mathrm{K}\{\mathrm{J}/a\}$. By Lemma 3.6, one obtains $\phi; \Phi\{\mathrm{J}/a\}, \Psi \vdash^{\mathcal{E}} (\mu\{\mathbf{0}/b\})\{\mathrm{J}/a\} \sqsubseteq \sigma\{\mathrm{J}/a\}$. Please observe that b can be assumed not to occur free in J, and as a consequence $(\mu\{\mathbf{0}/b\})\{\mathrm{J}/a\} \equiv (\mu\{\mathrm{J}/a\})\{\mathbf{0}/b\}$. Similarly, $\phi; \Phi\{\mathrm{J}/a\}, \Psi \vdash^{\mathcal{E}} (([b < \mathrm{K}] \cdot \mu)\{\mathrm{J}/a\}) \Downarrow$ and $\phi; \Phi\{\mathrm{J}/a\}, \Psi \vdash^{\mathcal{E}} \Gamma\{\mathrm{J}/a\} \Downarrow$. Again, $([b < \mathrm{K}] \cdot \mu)\{\mathrm{J}/a\}$ is syntactically identical to $[b < \mathrm{K}\{\mathrm{J}/a\}] \cdot \mu\{\mathrm{J}/a\}$. As a consequence:
$$\frac{\begin{array}{c}\phi; \Phi\{\mathrm{J}/a\}, \Psi \models^{\mathcal{E}} \mathbf{0} \leqslant \mathrm{I}\{\mathrm{J}/a\} \qquad \phi; \Phi\{\mathrm{J}/a\}, \Psi \models^{\mathcal{E}} \mathbf{1} \leqslant \mathrm{K}\{\mathrm{J}/a\} \\ \phi; \Phi\{\mathrm{J}/a\}, \Psi \vdash^{\mathcal{E}} (\mu\{\mathrm{J}/a\})\{\mathbf{0}/b\} \sqsubseteq \sigma\{\mathrm{J}/a\} \\ \phi; \Phi\{\mathrm{J}/a\}, \Psi \vdash^{\mathcal{E}} ([b < \mathrm{K}\{\mathrm{J}/a\}] \cdot \mu\{\mathrm{J}/a\}) \Downarrow \qquad \phi; \Phi\{\mathrm{J}/a\}, \Psi \vdash^{\mathcal{E}} (\Gamma\{\mathrm{J}/a\}) \Downarrow\end{array}}{\phi; \Phi\{\mathrm{J}/a\}, \Psi; \Gamma\{\mathrm{J}/a\}, x : [b < \mathrm{K}\{\mathrm{J}/a\}] \cdot \mu\{\mathrm{J}/a\} \vdash^{\mathcal{E}}_{\mathrm{I}\{\mathrm{J}/a\}} x : \sigma\{\mathrm{J}/a\}}\ V$$
- If π is
$$\frac{\phi, a; \Phi; \Gamma, x : [b < \mathrm{K}] \cdot \mu \vdash_{\mathrm{I}} t : \tau}{\phi, a; \Phi; \Gamma \vdash_{\mathrm{I}} \lambda x.t : [b < \mathrm{K}] \cdot \mu \multimap \tau}\ L$$
then, by the induction hypothesis we get
$$\phi; \Phi\{\mathrm{J}/a\}, \Psi; \Gamma\{\mathrm{J}/a\}, x : [b < \mathrm{K}\{\mathrm{J}/a\}] \cdot \mu\{\mathrm{J}/a\} \vdash_{\mathrm{I}\{\mathrm{J}/a\}} t : \tau\{\mathrm{J}/a\}.$$
As a consequence, we can conclude by
$$\frac{\phi; \Phi\{\mathrm{J}/a\}, \Psi; \Gamma\{\mathrm{J}/a\}, x : [b < \mathrm{K}\{\mathrm{J}/a\}] \cdot \mu\{\mathrm{J}/a\} \vdash_{\mathrm{I}\{\mathrm{J}/a\}} t : \tau\{\mathrm{J}/a\}}{\phi; \Phi\{\mathrm{J}/a\}, \Psi; \Gamma\{\mathrm{J}/a\} \vdash_{\mathrm{I}\{\mathrm{J}/a\}} \lambda x.t : ([b < \mathrm{K}] \cdot \mu \multimap \tau)\{\mathrm{J}/a\}}\ L$$
since $[b < \mathrm{K}\{\mathrm{J}/a\}] \cdot \mu\{\mathrm{J}/a\} \multimap \tau\{\mathrm{J}/a\} \equiv ([b < \mathrm{K}] \cdot \mu \multimap \tau)\{\mathrm{J}/a\}$.

The other cases are similar. □

A particularly useful instance of Lemma 3.9 is the following:

Lemma 3.10 (Instantiation)**.** *Let* $\phi, a; \Phi, a < \mathrm{I} \vdash_{\mathrm{K}} t : \sigma$*. If* $\phi; \Psi \models_{\mathcal{E}} \mathrm{J} < \mathrm{I}$*, then,* $\phi; \Phi\{\mathrm{J}/a\}, \Psi \vdash_{\mathrm{K}\{\mathrm{J}/a\}} t : \sigma\{\mathrm{J}/a\}$*.*

Proof. By Lemma 3.9 and Lemma 3.7. □

Moreover a Generation Lemma will be useful.

Lemma 3.11 (Generation)**.**
1. *Let* $\phi;\Phi;\Gamma \vdash_{\mathrm{K}} \lambda x.t : \sigma$*, then* $\sigma = [a < \mathrm{I}] \cdot \tau \multimap \mu$ *and* $\phi;\Phi;\Gamma, x : [a < \mathrm{I}] \cdot \tau \vdash_{\mathrm{K}} t : \mu$*;*
2. *Let* $\phi;\Phi;\Gamma \vdash_{\mathrm{K}} \underline{0} : \mathtt{Nat}[\mathrm{I},\mathrm{J}]$*, then* $\phi;\Phi \models^{\mathcal{E}} \mathrm{I} = \boldsymbol{0}$*;*
3. *Let* $\phi;\Phi;\Gamma \vdash_{\mathrm{K}} \underline{\mathtt{n}+\mathtt{1}} : \mathtt{Nat}[\mathrm{I},\mathrm{J}]$*, then* $\phi;\Phi \models^{\mathcal{E}} \mathrm{J} \geqslant \boldsymbol{1}$*.*

Proof. All the points are immediate by an inspection of the rules. □

We are now ready to embark on a proof of Subject Reduction. As usual, the first step is a Substitution Lemma:

Lemma 3.12 (Term Substitution)**.** *Let* $\phi, a;\Phi, a < \mathrm{I};\varnothing \vdash_{\mathrm{J}} t : \sigma$ *and* $\phi;\Phi;x : [a < \mathrm{I}] \cdot \sigma, \Delta \vdash_{\mathrm{K}} u : \tau$*. Then we have* $\phi;\Phi;\Delta \vdash_{\mathrm{H}} u\{t/x\} : \tau$ *for some* H *such that* $\phi;\Phi \models^{\mathcal{E}} \mathrm{H} \leqslant \mathrm{K} + \mathrm{I} + \sum_{a<\mathrm{I}} \mathrm{J}$*.*

Proof. As usual, this is an induction on the structure of a type derivation for u. All relevant inductive cases require some manipulation of the type derivation for t. The previous lemmas give exactly the right degree of "malleability". Let π be a derivation for

$$\phi;\Phi;x : [a < \mathrm{I}] \cdot \sigma, \Delta \vdash_{\mathrm{K}} u : \tau.$$

Let us examine some interesting cases, dependently on the shape of π:

- Consider π to be just

$$\frac{\begin{array}{c}\phi;\Phi \models^{\mathcal{E}} \mathbf{0} \leqslant \mathrm{K} \qquad \phi;\Phi \models^{\mathcal{E}} \mathbf{1} \leqslant \mathrm{I} \\ \phi;\Phi \vdash^{\mathcal{E}} \sigma\{\mathbf{0}/a\} \sqsubseteq \tau \\ \phi;\Phi \vdash^{\mathcal{E}} ([a < \mathrm{I}] \cdot \sigma) \Downarrow \qquad \phi;\Phi \vdash^{\mathcal{E}} \Delta \Downarrow\end{array}}{\phi;\Phi;\Delta, x : [a < \mathrm{I}] \cdot \sigma \vdash_{\mathrm{K}} x : \tau}\ V$$

  Since $\phi;\varnothing \models \mathbf{0} \Downarrow$, applying Lemma 3.10 we have

$$\phi;\Phi\{\mathbf{0}/a\};\varnothing \vdash_{\mathrm{J}\{\mathbf{0}/a\}} t : \sigma\{\mathbf{0}/a\}$$

  and since Φ does not contain free occurrences of a we obtain:

$$\phi;\Phi;\varnothing \vdash_{\mathrm{J}\{\mathbf{0}/a\}} t : \sigma\{\mathbf{0}/a\}.$$

  Now, by applying Lemma 3.8, Lemma 3.5 and Lemma 3.7 we can conclude

$$\phi;\Phi;\Delta \vdash_{\mathrm{K}+\mathrm{I}+\sum_{a<\mathrm{I}} \mathrm{J}} t : \tau$$

  since clearly

$$\phi;\Phi \models \mathrm{J}\{\mathbf{0}/a\} \leqslant \mathrm{K} + \mathrm{I} + \sum_{a<\mathrm{I}} \mathrm{J}.$$

- Let us consider the case π ends by an instance of the A rule. In particular, without loss of generality we can consider a situation as the following:

$$\frac{\begin{array}{c}\phi;\Phi;x : [a < \mathrm{K}] \cdot \gamma \vdash_{\mathrm{L}} v : [b < \mathrm{N}] \cdot \mu \multimap \tau \\ \phi, b;\Phi, b < \mathrm{N};x : [a < \mathrm{H}] \cdot (\gamma\{\mathrm{K} + a + \sum_{d<b} \mathrm{H}\{d/b\}/a\}) \vdash_{\mathrm{M}} u : \mu \\ \phi;\Phi \vdash [a < \mathrm{I}] \cdot \sigma \sqsubseteq [a < \mathrm{K} + \sum_{b<\mathrm{N}} \mathrm{H}] \cdot \gamma \\ \phi;\Phi \models \mathrm{Q} \geqslant \mathrm{L} + \mathrm{N} + \sum_{b<\mathrm{N}} \mathrm{M}\end{array}}{\phi;\Phi;x : [a < \mathrm{I}] \cdot \sigma \vdash_{\mathrm{Q}} vu : \tau}\ A$$

  By definition of subtyping, $\phi;\Phi, a < \mathrm{I} \vdash \sigma \sqsubseteq \gamma$, and $\phi;\Phi \models^{\mathcal{E}} \mathrm{K} + \mathrm{P} \leqslant \mathrm{I}$, where $\mathrm{P} \equiv \sum_{b<\mathrm{N}} \mathrm{H}$. So, by Lemma 3.4, we have

$$\phi;\Phi, a < \mathrm{K} + \mathrm{P};\varnothing \vdash_{\mathrm{J}} t : \sigma$$

and by Lemma 3.7 we have

$$\phi; \Phi, a < \mathrm{K} + \mathrm{P}; \varnothing \vdash_{\mathrm{J}} t : \gamma$$

(since $\phi; \Phi, a < \mathrm{K} + \mathrm{P} \vdash \sigma \sqsubseteq \gamma$). Applying again Lemma 3.4 we obtain

$$\phi; \Phi, a < \mathrm{K}; \varnothing \vdash_{\mathrm{J}} t : \gamma$$

and by induction hypothesis we get

$$\phi; \Phi; \varnothing \vdash_{\mathrm{T}} v\{t/x\} : [b < \mathrm{N}] \cdot \mu \multimap \tau$$

with $\phi; \Phi \models^{\mathcal{E}} \mathrm{T} \leqslant \mathrm{L} + \mathrm{K} + \sum_{a<\mathrm{K}} \mathrm{J}$. We observe that

$$\phi, b, c; \Phi, a \leqslant \mathrm{K} + c + \sum_{d<b} \mathrm{H}\{d/b\}, b < \mathrm{N}, c < \mathrm{H} \models^{\mathcal{E}} a < \mathrm{K} + \mathrm{P}.$$

By Lemma 3.4 we get

$$\phi, b, c; \Phi, a \leqslant \mathrm{K} + c + \sum_{d<b} \mathrm{H}\{d/b\}, b < \mathrm{N}, c < \mathrm{H}; \varnothing \vdash_{\mathrm{J}} t : \gamma$$

and by Lemma 3.9 and Lemma 3.7 we obtain

$$\phi; \Phi, a < \mathrm{H}, b < \mathrm{N}; \varnothing \vdash_{\mathrm{R}} t : \gamma\{\mathrm{K} + a + \sum_{d<b} \mathrm{H}\{d/b\}/a\},$$

where $\mathrm{R} \equiv \mathrm{J}\{\mathrm{K} + a + \sum_{d<b} \mathrm{H}\{d/b\}/a\}$. By induction hypothesis, we get

$$\phi; \Phi, b < \mathrm{N}; \varnothing \vdash_{\mathrm{S}} u\{t/x\} : \mu$$

with $\phi; \Phi \models^{\mathcal{E}} \mathrm{S} \leqslant \mathrm{M} + \mathrm{H} + \sum_{a<\mathrm{H}} \mathrm{R}$. And we can conclude as follows:

$$\frac{\phi; \Phi; \varnothing \vdash_{\mathrm{T}} v\{t/x\} : [b < \mathrm{N}] \cdot \mu \multimap \tau \qquad \phi; \Phi; \varnothing \vdash_{\mathrm{S}} u\{t/x\} : \mu}{\phi; \Phi; \varnothing \vdash_{\mathrm{T}+\mathrm{N}+\sum_{b<\mathrm{J}} \mathrm{S}} v\{t/x\}u\{t/x\} : \tau}\ A$$

Please observe that:

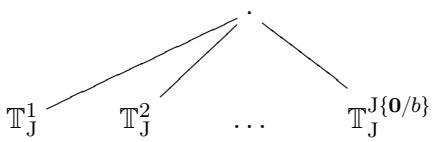
$$\begin{aligned}
\phi; \Phi \models^{\mathcal{E}} \mathrm{T} + \mathrm{N} + \sum_{b<\mathrm{J}} \mathrm{S} &\leqslant (\mathrm{L} + \mathrm{K} + \sum_{a<\mathrm{K}} \mathrm{J}) + \mathrm{N} + \sum_{b<\mathrm{N}} (\mathrm{M} + \mathrm{H} + \sum_{a<\mathrm{H}} \mathrm{R}) \\
&\leqslant (\mathrm{L} + \mathrm{N} + \sum_{b<\mathrm{N}} \mathrm{M}) + (\mathrm{K} + \sum_{b<\mathrm{N}} \mathrm{H}) + (\sum_{a<\mathrm{K}} \mathrm{J} + \sum_{b<\mathrm{N}} \sum_{a<\mathrm{H}} \mathrm{R}) \\
&\leqslant (\mathrm{L} + \mathrm{N} + \sum_{b<\mathrm{N}} \mathrm{M}) + (\mathrm{K} + \sum_{b<\mathrm{N}} \mathrm{H}) + \sum_{a<\mathrm{K}+\sum_{b<\mathrm{N}} \mathrm{H}} \mathrm{J} \\
&\leqslant \mathrm{Q} + \mathrm{I} + \sum_{a<\mathrm{I}} \mathrm{J}.
\end{aligned}$$

The other cases are similar. □

Theorem 3.13 (Subject Reduction)**.** *Let* $\phi;\Phi;\varnothing \vdash_{\mathrm{I}} t : \sigma$ *and* $t \to u$*. Then,* $\phi;\Phi;\varnothing \vdash_{\mathrm{J}} u : \sigma$*, where* $\phi;\Phi \models \mathrm{J} \leqslant \mathrm{I}$*.*

Proof. By induction on the structure of a derivation π for $\phi;\Phi;\varnothing \vdash_{\mathrm{I}} t : \sigma$ Let us examine the distinct cases:

- Suppose π is
$$\dfrac{\begin{array}{c}\phi;\Phi;\varnothing \vdash_{\mathrm{K}} \lambda x.t : [a < \mathrm{H}] \cdot \tau \multimap \sigma \\ \phi;\Phi, a < \mathrm{H};\varnothing \vdash_{\mathrm{L}} u : \tau \\ \phi;\Phi \models \mathrm{K} + \mathrm{H} + \sum_{a<\mathrm{H}} \mathrm{L} \leqslant \mathrm{I}\end{array}}{\phi;\Phi;\varnothing \vdash_{\mathrm{I}} (\lambda x.t)u : \sigma}\ A$$
By Lemma 3.11, Point 1, we have $\phi;\Phi;x : [a < \mathrm{H}] \cdot \tau \vdash_{\mathrm{K}} t : \sigma$. Then by Lemma 3.12 we can conclude:
$$\phi;\Phi;\varnothing \vdash_{\mathrm{J}} t\{u/x\} : \sigma$$
for $\phi;\Phi \models^{\mathcal{E}} \mathrm{J} \leqslant \mathrm{K} + \mathrm{H} + \sum_{a<\mathrm{H}} \mathrm{L} \leqslant \mathrm{I}$.
- Suppose π is
$$\dfrac{\begin{array}{c}\phi;\Phi;\varnothing \vdash_{\mathrm{K}} \underline{\mathtt{0}} : \mathtt{Nat}[\mathrm{K},\mathrm{H}] \\ \phi;\Phi, \mathrm{H} \leqslant \mathtt{0};\varnothing \vdash_{\mathrm{L}} v : \sigma \\ \phi;\Phi, \mathrm{K} \geqslant 1;\varnothing \vdash_{\mathrm{L}} w : \sigma \\ \phi;\Phi \models \mathrm{K} + \mathrm{L} \leqslant \mathrm{I}\end{array}}{\phi;\Phi;\varnothing \vdash_{\mathrm{I}} \ \mathtt{ifz}\ \underline{\mathtt{0}}\ \mathtt{then}\ v\ \mathtt{else}\ w : \sigma}\ F$$
By Lemma 3.11, Point 2, we have $\phi;\Phi \models^{\mathcal{E}} \mathrm{H} \leqslant \mathtt{0}$. So, by Lemma 3.4 we can conclude $\phi;\Phi;\varnothing \vdash_{\mathrm{L}} v : \sigma$.
- Suppose π is
$$\dfrac{\begin{array}{c}\phi;\Phi;\varnothing \vdash_{\mathrm{K}} \underline{\mathtt{n}+\mathtt{1}} : \mathtt{Nat}[\mathrm{K},\mathrm{H}] \\ \phi;\Phi, \mathrm{H} \leqslant \mathtt{0};\varnothing \vdash_{\mathrm{L}} v : \sigma \\ \phi;\Phi, \mathrm{K} \geqslant 1;\varnothing \vdash_{\mathrm{L}} w : \sigma \\ \phi;\Phi \models \mathrm{K} + \mathrm{L} \leqslant \mathrm{I}\end{array}}{\phi;\Phi;\varnothing \vdash_{\mathrm{I}} \ \mathtt{ifz}\ \underline{\mathtt{n}+\mathtt{1}}\ \mathtt{then}\ v\ \mathtt{else}\ w : \sigma}\ F$$
By Lemma 3.11, Point 3, we have $\phi;\Phi \models^{\mathcal{E}} \mathrm{K} \geqslant 1$. So, by Lemma 3.4 we have $\phi;\Phi;\varnothing \vdash_{\mathrm{L}} w : \sigma$.
- Suppose π is
$$\dfrac{\begin{array}{c}\phi;\Phi \vdash \triangle_b^{\mathbf{0},\mathbf{1}} \mathrm{J} \leqslant \mathrm{L}, \mathrm{P} \\ \phi, b;\Phi, b < \mathrm{L};x : [a < \mathrm{J}] \cdot \mu \vdash_{\mathrm{K}} t : \tau \\ \phi;\Phi \vdash \tau\{\mathbf{0}/b\} \sqsubseteq \sigma \\ \phi, a, b;\Phi, a < \mathrm{J}, b < \mathrm{L} \vdash \tau\{\triangle_b^{b+1,a} \mathrm{J} + b + 1/b\} \sqsubseteq \mu \\ \phi, \Phi \models \mathrm{P} \dot{-} 1 + \sum_{b<\mathrm{L}} \mathrm{K} \leqslant \mathrm{I}\end{array}}{\phi;\Phi;\varnothing \vdash_{\mathrm{I}} \mathtt{fix}\ x.t : \sigma}\ R$$
The index term J describes a tree $\mathbb{T}_{\mathrm{J}}$ (in the sense of forest cardinalities, see Section 3.1) which in turn represents the tree of recursive calls. $\mathbb{T}_{\mathrm{J}}$ looks as follows:



where $\mathbb{T}_{\mathrm{J}}^i$ represents the tree of recursive calls triggered by the i-th call to x in t. We first proceed by giving a type to t which somehow corresponds to the root of $\mathbb{T}_{\mathrm{J}}$. This will be done by substituting b for $\mathbf{0}$ in the derivation we get as an hypothesis of π. Since $\phi; \Phi \models_{\mathcal{E}} \mathbf{0} < \mathrm{L}$, by Lemma 3.10 we have

$$\phi; \Phi; x : [a < \mathrm{J}\{\mathbf{0}/b\}] \cdot \sigma\{\mathbf{0}/b\} \vdash_{\mathrm{K}\{\mathbf{0}/b\}} t : \tau\{\mathbf{0}/b\}.$$

From the hypothesis $\phi; \Phi \vdash \tau\{\mathbf{0}/b\} \sqsubseteq \sigma$ and by the Subtyping Lemma, we obtain

$$\phi; \Phi; x : [a < \mathrm{J}\{\mathbf{0}/b\}] \cdot \sigma\{\mathbf{0}/b\} \vdash_{\mathrm{K}\{\mathbf{0}/b\}} t : \sigma.$$

Our objective now is building *one* type derivation for $\mathtt{fix}\ x.t$ that somehow reflect the $\mathrm{J}\{\mathbf{0}/b\}$ subtrees $\mathbb{T}_{\mathrm{J}}^1, \ldots, \mathbb{T}_{\mathrm{J}}^{\mathrm{J}\{\mathbf{0}/b\}}$. Speaking more formally, we want to prove that:

$$\phi; \Phi, a < \mathrm{J}\{\mathbf{0}/b\} \vdash_{\mathrm{R}} \mathtt{fix}\ x.t : \sigma\{\mathbf{0}/b\} \tag{3.5}$$

where

$$\phi; \Phi \models \mathrm{K}\{\mathbf{0}/b\} + \mathrm{J}\{\mathbf{0}/b\} + \sum_{a < \mathrm{J}\{\mathbf{0}/b\}} \mathrm{R} \leqslant \mathrm{I}.$$

That would immediately lead to the thesis. To reach (3.5), we proceed by first defining two index terms with a quite intuitive informal semantics:

- First of all, we define M to be $\triangle_b^{\mathbf{0},\mathbf{1}} \mathrm{J}\{b + \mathbf{1} + \triangle_b^{\mathbf{1},c} \mathrm{J}/b\}$. Observe that c occurs free in M; indeed, M counts the number of nodes in the tree $\mathbb{T}_{\mathrm{J}}^c$.
- Another useful index term is N, which is defined to be $\mathbf{1} + b + \sum_{c<a} \mathrm{M}$. N is designed as to return the label of a node in $\mathbb{T}_{\mathrm{J}}^a$ given a and the offset b. In other words, $\mathbb{T}_{\mathrm{J}\{\mathrm{N}/b\}}$ is a recursion tree isomorphic to $\mathbb{T}_{\mathrm{J}}^a$.

Now, if we substitute b for N in one of the premises of π, we get

$$\phi, a, b; \Phi, a < \mathrm{J}\{\mathbf{0}/b\}, b < \mathrm{M}\{a/c\}; x : [d < \mathrm{J}\{\mathrm{N}/b\}] \cdot \mu\{d/a\}\{\mathrm{N}/b\} \vdash_{\mathrm{K}\{\mathrm{N}/b\}} t : \tau\{\mathrm{N}/b\}. \tag{3.6}$$

Since by Lemma 3.2 we have $\sum_{c<e} \mathrm{M} \simeq \triangle_b^{\mathbf{1},e} \mathrm{J}$ we know that

$$\overset{\mathbf{0},\mathbf{1}}{\underset{b}{\triangle}} \mathrm{J}\{\mathrm{N}/b\} \simeq \overset{\mathbf{0},\mathbf{1}}{\underset{b}{\triangle}} \mathrm{J}\{\mathbf{1} + b + \sum_{c<a} \mathrm{M}/b\} \simeq \overset{\mathbf{0},\mathbf{1}}{\underset{b}{\triangle}} \mathrm{J}\{\mathbf{1} + b + \overset{\mathbf{1},a}{\underset{b}{\triangle}} \mathrm{J}/b\} \simeq \mathrm{M}\{a/c\}. \tag{3.7}$$

Now, consider the problem of determining the index (in $\mathbb{T}_{\mathrm{J}}$) of the $(d+1)$-th children of a node of index b inside $\mathbb{T}_{\mathrm{J}}^a$. There are two equivalent ways to compute it:

- either you start from N, but then you substitute b by $b + \mathbf{1} + \triangle_b^{b+\mathbf{1},d} \mathrm{J}\{\mathrm{N}/b\}$;
- or you simply consider $\mathrm{N} + \mathbf{1} + \triangle_b^{\mathrm{N}+\mathbf{1},d} \mathrm{J}$.

In the first case, you compute the desired index by merely instantiating N appropriately, while in the second case you use N without altering it. The observation above can be formalized as follows:

$$\phi, a, b, d; \Phi, a < \mathrm{J}\{\mathbf{0}/b\}, b < \mathrm{M}\{a/c\}, d < \mathrm{J}\{\mathrm{N}/b\} \vdash \tau\{\mathrm{N}/b\}\{b + \mathbf{1} + \overset{b+\mathbf{1},d}{\underset{b}{\triangle}} \mathrm{J}\{\mathrm{N}/b\}/b\} \simeq \tau\{\mathrm{N} + \mathbf{1} + \overset{\mathrm{N}+\mathbf{1},d}{\underset{b}{\triangle}} \mathrm{J}/b\}.$$

By Lemma 3.10, we also obtain:

$$\phi, a, b, d; \Phi, a < \mathrm{J}\{\mathbf{0}/b\}, b < \mathrm{M}\{a/c\}, d < \mathrm{J}\{\mathrm{N}/b\} \vdash \tau\{\mathrm{N}/b\}\{b + \mathbf{1} + \bigcirc\!\!\!\!\!\!\triangle_{b}^{b+\mathbf{1},d} \mathrm{J}\{\mathrm{N}/b\}/b\} \sqsubseteq \sigma\{d/a\}\{\mathrm{N}/b\}. \quad (3.8)$$

Now, (3.6), (3.7) and (3.8) can be put together by way of rule R, and one then conclude that

$$\phi; \Phi, a < \mathrm{J}\{\mathbf{0}/b\}; \varnothing \vdash_{\mathrm{M}\{a/c\} \dot{-} \mathbf{1} + \sum_{b<\mathrm{M}\{a/c\}} \mathrm{K}\{\mathrm{N}/b\}} \mathtt{fix}\ x.t : \tau\{\mathrm{N}\{\mathbf{0}/b\}/b\}.$$

But instantiating one of the hypothesis' of π, we obtain

$$\phi, a; \Phi, a < \mathrm{J}\{\mathbf{0}/b\} \vdash \tau\{\bigcirc\!\!\!\!\!\!\triangle_{b}^{\mathbf{1},a} \mathrm{J} + \mathbf{1}/b\} \sqsubseteq \mu\{\mathbf{0}/b\}.$$

By Lemma 3.2, we can prove that $\bigcirc\!\!\!\!\!\!\triangle_{b}^{\mathbf{1},a} \mathrm{J} + \mathbf{1} = \mathrm{N}\{\mathbf{0}/b\}$. Indeed, this is quite intuitive: the index of the root of $\mathbb{T}^a_{\mathrm{J}}$ can be computed in two equivalent ways through J or through N. As a consequence,

$$\phi; \Phi, a < \mathrm{J}\{\mathbf{0}/b\}; \varnothing \vdash_{\mathrm{R}} \mathtt{fix}\ x.t : \sigma\{\mathbf{0}/b\},$$

where $\mathrm{R} \equiv \mathrm{M}\{a/c\} \dot{-} \mathbf{1} + \sum_{b<\mathrm{M}\{a/c\}} \mathrm{K}\{\mathrm{N}/b\}$. But we are done, since

$$\begin{aligned}
\phi; \Phi \models\ & \mathrm{K}\{\mathbf{0}/b\} + \mathrm{J}\{\mathbf{0}/b\} + \sum_{a<\mathrm{J}\{\mathbf{0}/b\}} \mathrm{R} \\
& \equiv \mathrm{K}\{\mathbf{0}/b\} + \mathrm{J}\{\mathbf{0}/b\} + \sum_{a<\mathrm{J}\{\mathbf{0}/b\}} (\mathrm{M}\{a/c\} \dot{-} \mathbf{1} + \sum_{b<\mathrm{M}\{a/c\}} \mathrm{K}\{\mathrm{N}/b\}) \\
& = (\mathrm{J}\{\mathbf{0}/b\} + \sum_{a<\mathrm{J}\{\mathbf{0}/b\}} (\mathrm{M}\{a/c\} \dot{-} \mathbf{1})) + \mathrm{K}\{\mathbf{0}/b\} + \sum_{a<\mathrm{J}\{\mathbf{0}/b\}} \sum_{b<\mathrm{M}\{a/c\}} \mathrm{K}\{\mathrm{N}/b\} \\
& \leqslant \bigcirc\!\!\!\!\!\!\triangle_{b}^{\mathbf{1},\mathrm{J}\{\mathbf{0}/b\}} \mathrm{J} + \mathrm{K}\{\mathbf{0}/b\} + \sum_{a<\mathrm{J}\{\mathbf{0}/b\}} \sum_{b<\mathrm{M}\{a/c\}} \mathrm{K}\{\mathrm{N}/b\} \\
& \leqslant \mathrm{P} \dot{-} \mathbf{1} + \sum_{b<\mathrm{L}} \mathrm{K} \leqslant \mathrm{I}.
\end{aligned}$$

This concludes the proof. □

4. Intensional Soundness

Subject Reduction already implies an *extensional* notion of soundness for programs: if a term t can be typed with $\vdash_{\mathrm{K}} t : \mathtt{Nat}[\mathrm{I}, \mathrm{J}]$, then its normal form (if any) is a natural number between $[\![\mathrm{I}]\!]$ and $[\![\mathrm{J}]\!]$. However, Subject Reduction does not tell us whether the evaluation of t terminates, and in how much time. Has K anything to do with the complexity of evaluating t? The only information that can be extracted from the Subject Reduction Theorem is that K does not increase along reduction.

In this section, *Intensional Soundness* (Theorem 4.6 below) for the type system $\mathsf{d}\ell\mathsf{PCF}$ will be proved. A Krivine's Machine $\mathsf{K}_{\mathsf{PCF}}$ for PCF programs will be first defined in Section 4.1. Given a program (i.e. a closed term of base type), the machine $\mathsf{K}_{\mathsf{PCF}}$ either evaluates it to normal form or diverges. A formal connection between the machine $\mathsf{K}_{\mathsf{PCF}}$ and the type

Term	Environment	Stack		Term	Environment	Stack
tu	ρ	ξ	$\to$	t	ρ	$(u,\rho)\cdot\xi$
$\lambda x.t$	ρ	$c\cdot\xi$	$\to$	t	$c\cdot\rho$	ξ
x	$(t_0,\rho_0)\cdots(t_n,\rho_n)$	ξ	$\to$	t_x	ρ_x	ξ
$\mathtt{ifz}\ t\ \mathtt{then}\ u\ \mathtt{else}\ v$	ρ	ξ	$\to$	t	ρ	$(u,v,\rho)\cdot\xi$
$\mathtt{fix}\ x.t$	ρ	ξ	$\to$	t	$(\mathtt{fix}\ x.t,\rho)\cdot\rho$	ξ
$\underline{\mathtt{n}}$	ρ	$\mathtt{s}\cdot\xi$	$\to$	$\underline{\mathtt{n+1}}$	ρ	ξ
$\underline{\mathtt{n}}$	ρ	$\mathtt{p}\cdot\xi$	$\to$	$\underline{\mathtt{n\dot{-}1}}$	ρ	ξ
$\underline{\mathtt{0}}$	ρ	$(t,u,\mu)\cdot\xi$	$\to$	t	μ	ξ
$\underline{\mathtt{n+1}}$	ρ	$(t,u,\mu)\cdot\xi$	$\to$	u	μ	ξ
$\mathtt{s}(t)$	ρ	ξ	$\to$	t	ρ	$\mathtt{s}\cdot\xi$
$\mathtt{p}(t)$	ρ	ξ	$\to$	t	ρ	$\mathtt{p}\cdot\xi$

Figure 7: The $\mathsf{K_{PCF}}$ Transition Steps.

system $\mathsf{d\ell PCF}$ will be established by means of a weighted typability notion for machine configurations, introduced in Section 4.2. This notion is the fundamental ingredient to keep track of the number of machine steps.

4.1. **The $\mathsf{K_{PCF}}$ Machine.** The Krivine's Machine has been introduced as a natural device to evaluate pure lambda-terms under a weak-head notion of reduction [27]. Here, the standard Krivine's Machine is extended to a machine $\mathsf{K_{PCF}}$ which handles not only abstractions and applications, but also constants, conditionals and fixpoints.

The *configurations* of the machine $\mathsf{K_{PCF}}$, ranged over by $C, D, \ldots$, are triples $C = (t, \rho, \xi)$ where ρ and ξ are two additional constructions: ρ is an *environment*, that is a (possibly empty) finite sequence of *closures*; while ξ is a (possibly empty) *stack* of *contexts*. Stacks are ranged over by $\xi, \theta, \ldots$. A *closure*, as usual, is a pair $c = (t, \rho)$ where t is a term and ρ is an environment. A *context* is either a closure, a term $\mathtt{s}$, a term $\mathtt{p}$, or a triple (u, v, ρ) where u, v are terms and ρ is an environment.

The transition steps between configurations of the $\mathsf{K_{PCF}}$ machine are given in Figure 7. The transition rules require some comments. First of all, a naïve management of name variables is used. A more effective description however, could be given by using standard de Bruijn indexes. Note that the triple (u, v, ρ) is used as a context for the conditional construction; moreover, in a recursion step, a copy of the recursive term is put in a closure on the top of the current environment. As usual, the symbol $\to^*$ denotes the reflexive and transitive closure of the transition relation $\to$. The relation $\to^*$ implements weak-head reduction. Weak-head normal form and the normal form coincide for programs. So the machine $\mathsf{K_{PCF}}$ is a correct device to evaluate programs. For this reason, the notation $t \Downarrow \underline{\mathtt{n}}$ can be used as a shorthand for $(t, \varepsilon, \varepsilon) \to^* (\underline{\mathtt{n}}, \rho, \varepsilon)$. Moreover, notations like $C \Downarrow^n$ could also be used to stress that C reduces to an irreducible configuration in exactly n steps. The proof of the formal correctness of the abstract machine is outside the scope of this paper, however it should be clear that it could be obtained as a simple extension of the original one [27].

Intensional Soundness will be proved by studying how the weight I of any program t evolves during the evaluation of t by $\mathsf{K_{PCF}}$. This is possible because every reduction step in t is decomposed into a number of transitions in $\mathsf{K_{PCF}}$, and this decomposition highlights *when*, precisely, the weight changes. The same would be more difficult when performing plain reduction on terms. Proving Intensional Soundness this way requires, however, to keep

Closures

$$\frac{\phi;\Phi;x_1:[a<\mathrm{I}_1]\cdot\tau_1,\ldots,x_n:[a<\mathrm{I}_n]\cdot\tau_n\vdash_{\mathrm{K}} t:\sigma \quad \phi,a;\Phi,a<\mathrm{I}_i\vdash^{\mathcal{E}}_{\mathrm{H}_i} c_i:\tau_i \quad \phi;\Phi\models \mathrm{J}\geqslant \mathrm{K}+\mathrm{I}_1+\ldots+\mathrm{I}_n+\sum_{a<\mathrm{I}_1}\mathrm{H}_1+\ldots+\sum_{a<\mathrm{I}_n}\mathrm{H}_n.}{\phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{J}}(t,c_1\cdots c_n):\sigma}$$

Stacks

$$\frac{\phi;\Phi\models^{\mathcal{E}}\mathrm{J}\geqslant\mathbf{0} \quad \phi;\Phi\vdash^{\mathcal{E}}\sigma\sqsubseteq\tau}{\phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{J}}\varepsilon:(\sigma,\tau)} \qquad \frac{\phi;\Phi,a<\mathrm{I}\vdash^{\mathcal{E}}_{\mathrm{K}} c:\gamma \quad \phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{H}}\theta:(\mu,\tau) \quad \phi;\Phi\models^{\mathcal{E}}\mathrm{J}\geqslant\mathrm{H}+\sum_{a<\mathrm{I}}\mathrm{K}+\mathrm{I}}{\phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{J}} c\cdot\theta:([a<\mathrm{I}]\cdot\gamma\multimap\mu,\tau)}$$

$$\frac{\phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{J}}\theta:(\mathtt{Nat}[\mathrm{K},\mathrm{H}],\tau) \quad \phi;\Phi\vdash^{\mathcal{E}}\mathtt{Nat}[\mathrm{I}+1,\mathrm{L}+1]\sqsubseteq\mathtt{Nat}[\mathrm{K},\mathrm{H}]}{\phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{J}}\mathtt{s}\cdot\theta:(\mathtt{Nat}[\mathrm{I},\mathrm{L}],\tau)} \qquad \frac{\phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{J}}\theta:(\mathtt{Nat}[\mathrm{K},\mathrm{H}],\tau) \quad \phi;\Phi\vdash^{\mathcal{E}}\mathtt{Nat}[\mathrm{I}\dot{-}1,\mathrm{L}\dot{-}1]\sqsubseteq\mathtt{Nat}[\mathrm{K},\mathrm{H}]}{\phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{J}}\mathtt{p}\cdot\theta:(\mathtt{Nat}[\mathrm{I},\mathrm{L}],\tau)}$$

$$\frac{\phi;\Phi,\mathrm{I}\leqslant 0\vdash^{\mathcal{E}}_{\mathrm{K}}(t,\rho):\mu \quad \phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{H}}\theta:(\mu,\tau) \quad \phi;\Phi,\mathrm{L}\geqslant 1\vdash^{\mathcal{E}}_{\mathrm{K}}(u,\rho):\mu \quad \phi;\Phi\models^{\mathcal{E}}\mathrm{J}\geqslant\mathrm{K}+\mathrm{H}}{\phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{J}}(t,u,\rho)\cdot\theta:(\mathtt{Nat}[\mathrm{I},\mathrm{L}],\tau)}$$

Configurations

$$\frac{\phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{K}}(t,\rho):\sigma \quad \phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{J}}\xi:(\sigma,\tau) \quad \phi;\Phi\models^{\mathcal{E}}\mathrm{I}\geqslant\mathrm{K}+\mathrm{J}}{\phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{I}}(t,\rho,\xi):\tau}$$

Figure 8: Lifting $\mathsf{d}\ell\mathsf{PCF}$ Typing to Closures, Stacks and Configurations.

track of the types and weights of all objects in a machine configuration. In other words, the type system should be somehow generalized to an assignment system on *configurations*.

4.2. **Types and Weights for Configurations.** Assigning types and weights to configurations amounts to somehow keeping track of the nature of all terms appearing in environments and stacks. This is captured by the rules in Figure 8. A formal connection between typed terms and typed configurations could be established as expected, and such connection could be shown to be preserved by reduction. However, the following lemma is everything we need in the sequel:

Lemma 4.1. *Let $t\in\mathcal{P}$. Then, $\phi;\Phi;\varnothing\vdash^{\mathcal{E}}_{\mathrm{I}} t:\sigma$ if and only if $\phi;\Phi\vdash^{\mathcal{E}}_{\mathrm{I}}(t,\varepsilon,\varepsilon):\sigma$.*

Analogous notions of typability for closures, stacks and configurations can be given following the simpler type discipline of PCF proper. They can be obtained by simplifying those for $\mathsf{d}\ell\mathsf{PCF}$, see Figure 9. If $C\to D$ and π is a derivation of $\vdash C:\sigma$, then a derivation ρ of $\vdash D:\sigma$ can be easily obtained by manipulating π, and we write $\pi\to\rho$.

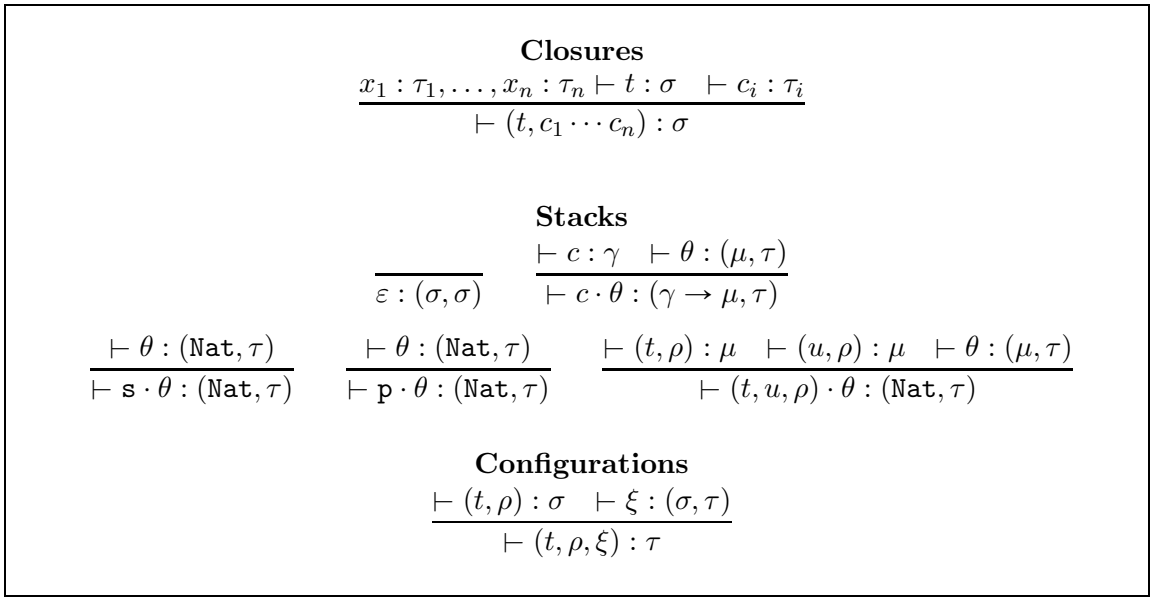


Figure 9: Extending PCF Typing to Closures, Stacks and Configurations.

4.3. **Measure Decreasing and Intensional Soundness.** An important property of Krivine's Machine says that during the evaluation of programs only subterms of the initial program are recorded in the environment. This justifies the notion of *size* for configurations, denoted $|C|$, that will be used in the sequel. This is defined as $|(t,\rho,\xi)| = |t| + |\xi|$. The size $|\xi|$ of a stack ξ is defined as the sum of sizes of its elements, where $|(t,\rho)| = |t|$, $|\mathtt{s}| = |\mathtt{p}| = 1$, and $|(t,u,\rho)| = |t| + |u|$. Moreover, another consequence of the same property is the following lemma.

Lemma 4.2. *Let $t \in \mathcal{P}$ and let $C = (t,\varepsilon,\varepsilon)$. Then, for each $D = (u,\rho,\xi)$ such that $C \to^* D$ and for each v occurring in ρ or ξ, $|v| \leqslant |t|$.*

Proof. Easy, by induction on the length of the reduction $C \to^* D$. In fact, a strengthening of the statement is needed for induction to work. In particular, not only $|v| \leqslant |t|$ for every v in ρ and ξ, but also for the *non-head subterms* of u. □

Intensional Soundness (Theorem 4.6) expresses the fact that for a program $t \in \mathcal{P}$ such that $\varnothing;\varnothing;\varnothing \vdash^{\mathcal{E}}_{\mathrm{I}} t : \mathtt{Nat}[\mathrm{J},\mathrm{K}]$, the number $[\![\mathrm{I}]\!]^{\mathcal{E}}_{\rho}$ is a good estimate of the number of steps needed to evaluate t. Moreover, thanks to Subject Reduction, the numbers $[\![\mathrm{J}]\!]^{\mathcal{E}}_{\rho}$ and $[\![\mathrm{K}]\!]^{\mathcal{E}}_{\rho}$ give an upper and a lower bound, respectively, to the result of such an evaluation. This is proved by showing that during reduction a measure, expressed as the combination of the weight and the size of a configuration, decreases. In turn, this requires extending some of the properties in Section 3.4 from terms to configurations. As an example, substitution holds on configurations, too:

Lemma 4.3. *If $\phi,a;\Phi \vdash^{\mathcal{E}}_{\mathrm{H}} (t,\rho):\sigma$, then $\phi;\Phi\{\mathrm{J}/a\},\Psi \vdash^{\mathcal{E}}_{\mathrm{H}\{\mathrm{J}/a\}} (t,\rho):\sigma\{\mathrm{J}/a\}$ for every J such that $\phi,\Psi \models^{\mathcal{E}} \mathrm{J} \Downarrow$.*

Proof. By induction on the proof of $\phi,a;\Phi \vdash^{\mathcal{E}}_{\mathrm{H}} (t,\rho):\sigma$, using Lemma 3.9. □

Moreover, type derivations for closures can be "split", exactly as terms:

Lemma 4.4. *Let* $\phi; \Phi \vdash^{\mathcal{E}} [a < \mathrm{I}] \cdot \sigma \sqsubseteq [a < \mathrm{J} + \mathrm{K}] \cdot \tau$ *and let* $\phi, a; \Phi, a < \mathrm{I} \vdash^{\mathcal{E}}_{\mathrm{H}} (t, \rho) : \sigma$ *Then, both* $(\phi, a; \Phi, a < \mathrm{J}) \vdash^{\mathcal{E}}_{\mathrm{H}} (t, \rho) : \tau$ *and* $\phi, a; \Phi, a < \mathrm{K} \vdash^{\mathcal{E}}_{\mathrm{H}\{\mathrm{J}+a/a\}} (t, \rho) : \tau\{\mathrm{J} + a/a\}$.

The key step towards Intensional Soundness is the following:

Lemma 4.5 (Weighted Subject Reduction)**.** *Suppose that* $(t, \varepsilon, \varepsilon) \to^* D \to E$ *and let* D *be such that* $\phi; \Phi \vdash^{\mathcal{E}}_{\mathrm{I}} D : \sigma$*. Then* $\phi; \Phi \vdash^{\mathcal{E}}_{\mathrm{J}} E : \sigma$*, and one of the following holds:*
1. $\phi; \Phi \models \mathrm{I} = \mathrm{J}$ *but* $|D| > |E|$*;*
2. $\phi; \Phi \models \mathrm{I} > \mathrm{J}$ *and* $|E| < |D| + |t|$.

Proof. The proof is by cases on the reduction $D \to E$. Condition 1 can be shown to apply to all the cases but the one in which $D = (x, \rho, \xi)$. In that one, weight decreasing relies on the side condition in the typing rule for variables, while the bound on the size increasing comes from Lemma 4.2. We just present some cases, the others can be obtained analogously:

- Consider the case $D \equiv (\ \texttt{ifz}\ w\ \texttt{then}\ u\ \texttt{else}\ v, \rho, \xi)$. We want to prove Point 1, namely that $E \equiv (t, \rho, (u, v, \rho) \cdot \xi)$ is such that $\phi; \Phi \vdash_{\mathrm{J}} E : \sigma$ where $\phi; \Phi \models \mathrm{I} = \mathrm{J}$ and $|D| > |E|$. The latter is immediate:

$$\begin{aligned} |D| &= 1 + |w| + |u| + |v| + |\xi| > |w| + (|u| + |v|) + |\xi| \\ &= |w| + |(u, v, \rho) \cdot \xi| = |E|. \end{aligned}$$

  Let us consider the former. By inspecting a proof of $\phi; \Phi \vdash^{\mathcal{E}}_{\mathrm{I}} D : \sigma$, we can easily derive the following judgments (where $\rho \equiv c_1, \ldots, c_n$):

$$\phi; \Phi; x_1 : [a < \mathrm{K}_1^w] \cdot \mu_1, \ldots, x_n : [a < \mathrm{K}_n^w] \cdot \mu_n \vdash_{\mathrm{I}_w} w : \mathtt{Nat}[\mathrm{H}, \mathrm{L}]; \tag{4.1}$$

$$\phi; \Phi, \mathrm{L} \leqslant \mathbf{0}; x_1 : [a < \mathrm{K}_1^{uv}] \cdot \mu_1\{\mathrm{K}_1^w + a/a\}, \ldots, x_n : [a < \mathrm{K}_n^{uv}] \cdot \mu_n\{\mathrm{K}_n^w + a/a\} \vdash_{\mathrm{I}_{uv}} u : \tau; \tag{4.2}$$

$$\phi; \Phi, \mathrm{H} \geqslant \mathbf{1}; x_1 : [a < \mathrm{K}_1^{uv}] \cdot \mu_1\{\mathrm{K}_1^w + a/a\}, \ldots, x_n : [a < \mathrm{K}_n^{uv}] \cdot \mu_n\{\mathrm{K}_n^w + a/a\} \vdash_{\mathrm{I}_{uv}} v : \tau; \tag{4.3}$$

$$\phi, a; \Phi, a < \mathrm{K}_i \vdash_{\mathrm{I}_{c_i}} c_i : \mu_i; \tag{4.4}$$

$$\phi; \Phi \vdash_{\mathrm{I}_\xi} \xi : (\tau, \sigma). \tag{4.5}$$

  where

$$\phi; \Phi \vdash [a < \mathrm{K}_i] \cdot \tau_i \sqsubseteq [a < \mathrm{K}_i^w] \cdot \mu_i \uplus [a < \mathrm{K}_i^{uv}] \cdot \mu_i\{\mathrm{K}_i^w + a/a\}; \tag{4.6}$$

$$\phi; \Phi \models \mathrm{I} \geqslant \mathrm{I}_w + \mathrm{I}_{uv} + \mathrm{K}_1 + \ldots + \mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{K}_n} \mathrm{I}_{c_n} + \mathrm{I}_\xi. \tag{4.7}$$

  By Lemma 4.4 applied to (4.4) and exploiting (4.6), we obtain that

$$\begin{aligned} &\phi, a; \Phi, a < \mathrm{K}_i^w \vdash_{\mathrm{I}_{c_i}} c_i : \mu_i; \\ &\phi, a; \Phi, a < \mathrm{K}_i^{uv} \vdash_{\mathrm{I}_{c_i}\{\mathrm{K}^w + a/a\}} c_i : \mu_i\{\mathrm{K}^w + a/a\}. \end{aligned}$$

  By way of (4.1), (4.2) and (4.3), we obtain

$$\begin{aligned} \phi; \Phi, \mathrm{L} \leqslant \mathtt{0} &\vdash_{\mathrm{I}_{(w,\rho)}} (w, \rho) : \mathtt{Nat}[\mathrm{H}, \mathrm{L}]; \\ \phi; \Phi, \mathrm{H} \geqslant \mathbf{1} &\vdash_{\mathrm{I}_{(uv,\rho)}} (u, \rho) : \tau; \\ \phi; \Phi &\vdash_{\mathrm{I}_{(uv,\rho)}} (v, \rho) : \tau; \end{aligned}$$

where

$$\mathrm{I}_{(w,\rho)} \equiv \mathrm{I}_w + \mathrm{K}_1^w + \ldots + \mathrm{K}_n^w + \sum_{a<\mathrm{K}_1^w} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{K}_n^w} \mathrm{I}_{c_n};$$

$$\mathrm{I}_{(uv,\rho)} \equiv \mathrm{I}_{uv} + \mathrm{K}_1^{uv} + \ldots + \mathrm{K}_n^{uv} + \sum_{a<\mathrm{K}_1^{uv}} \mathrm{I}_{c_1}\{\mathrm{K}_1^w + a/a\} \sum_{a<\mathrm{K}_n^{uv}} \mathrm{I}_{c_n}\{\mathrm{K}_n^w + a/a\}.$$

So, by definition and by (4.5) we have that $\phi;\Phi \vdash_{\mathrm{I}_{uv}+\mathrm{I}_\xi} (u,v,\rho)\cdot\xi : (\mathtt{Nat}[\mathrm{H},\mathrm{L}],t)$. Thus, we can conclude that $\phi;\Phi \vdash_{\mathrm{I}} E : \sigma$ (since from (4.7), it easily follows that $\phi;\Phi \models \mathrm{I} \geqslant \mathrm{I}_{(w,\rho)} + \mathrm{I}_{(uv,\rho)} + \mathrm{I}_\xi$).

- Consider the case $D \equiv (\lambda x.u, \rho, c\cdot\xi)$. We want to prove Point 1, namely that $E = (u, c\cdot\rho, \xi)$ is such that $\phi;\Phi \vdash_{\mathrm{J}} E : \sigma$ where $\phi;\Phi \models \mathrm{I} = \mathrm{J}$ and $|D| > |E|$. The latter is immediate, so let us consider the former. By inspecting a proof of $\phi;\Phi \vdash_{\mathrm{I}}^{\mathcal{E}} D : \sigma$, we can easily derive the following judgments (where $\rho \equiv c_1,\ldots,c_n$), in particular using the Generation Lemma:

$$\phi;\Phi;x_1 : [a<\mathrm{K}_1]\cdot\mu_1,\ldots,x_n : [a<\mathrm{K}_n]\cdot\mu_n, x : [a<\mathrm{H}]\cdot\gamma \vdash_{\mathrm{I}_u} u : \tau; \tag{4.8}$$

$$\phi,a;\Phi,a<\mathrm{K}_i \vdash_{\mathrm{I}_{c_i}} c_i : \mu_i; \tag{4.9}$$

$$\phi,a;\Phi,a<\mathrm{H} \vdash_{\mathrm{I}_c} c : \gamma; \tag{4.10}$$

$$\phi;\Phi \vdash_{\mathrm{I}_\xi} \xi : (\tau,\sigma). \tag{4.11}$$

Moreover:

$$\phi;\Phi \vdash \mathrm{I} \geqslant \mathrm{I}_u + \mathrm{K}_1 + \ldots + \mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{K}_n} \mathrm{I}_{c_n} + \mathrm{H} + \sum_{a<\mathrm{H}} \mathrm{I}_c + \mathrm{I}_\xi.$$

From (4.8), (4.9) and (4.10), we obtain $\phi;\Phi;\varnothing \vdash_{\mathrm{I}_{c\cdot\rho}} (u, c\cdot\rho) : \tau$, where

$$\mathrm{I}_{c\cdot\rho} \equiv \mathrm{I}_u + \mathrm{K}_1 + \ldots + \mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{K}_n} \mathrm{I}_{c_n} + \mathrm{H} + \sum_{a<\mathrm{H}} \mathrm{I}_c.$$

This, together with (4.11) easily yields the thesis.

- Consider the case $D \equiv (\underline{\mathtt{n}}, \rho, \mathtt{s}\cdot\xi)$. Again, we want to prove Point 1, that is $E = (\underline{\mathtt{n}+\mathtt{1}}, \rho, \xi)$ is such that $\phi;\Phi \vdash_{\mathrm{J}} E : \sigma$, where $\phi;\Phi \models \mathrm{I} = \mathrm{J}$ and $|D| > |E|$. The latter is easy:

$$|D| = |\underline{\mathtt{n}}| + |\mathtt{s}\cdot\xi| = 2 + |\xi| + 1 > 1 + |\xi| = |\underline{\mathtt{n}+\mathtt{1}}| + |\xi| = |E|,$$

so we consider the former. By inspecting a proof of $\phi;\Phi \vdash_{\mathrm{I}}^{\mathcal{E}} D : \sigma$, we can easily derive the following judgments (where $\rho \equiv c_1,\ldots,c_n$) in particular using the Generation Lemma:

$$\phi;\Phi;x_1 : [a<\mathrm{K}_1]\cdot\mu_1,\ldots,x_n : [a<\mathrm{K}_n]\cdot\mu_n \vdash_{\mathrm{I}_{\underline{\mathtt{n}}}} \underline{\mathtt{n}} : \mathtt{Nat}[\mathrm{H},\mathrm{L}]; \tag{4.12}$$

$$\phi,a;\Phi,a<\mathrm{K}_i \vdash_{\mathrm{I}_{c_i}} c_i : \mu_i; \tag{4.13}$$

$$\phi;\Phi \vdash_{\mathrm{I}_\xi} \xi : (\mathtt{Nat}[\mathrm{M},\mathrm{N}],\sigma). \tag{4.14}$$

Moreover:

$$\phi;\Phi \models \mathrm{I} \geqslant \mathrm{I}_{\underline{\mathtt{n}}} + \mathrm{K}_1 + \ldots + \mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{K}_n} \mathrm{I}_{c_n} + \mathrm{I}_\xi; \tag{4.15}$$

$$\phi;\Phi \vdash \mathtt{Nat}[\mathrm{H}+1,\mathrm{L}+1] \sqsubseteq \mathtt{Nat}[\mathrm{M},\mathrm{N}]. \tag{4.16}$$

From (4.12) and (4.16), we get

$$\phi;\Phi;x_1 : [a<\mathrm{K}_1]\cdot\mu_1,\ldots,x_n : [a<\mathrm{K}_n]\cdot\mu_n \vdash_{\mathrm{I}_{\underline{\mathtt{n}}}} \underline{\mathtt{n}+\mathtt{1}} : \mathtt{Nat}[\mathrm{M},\mathrm{N}].$$

This, together with (4.13), allows us to reach $\phi; \Phi \vdash_{\mathrm{I}_{(\underline{\mathtt{n+1}},\rho)}} (\underline{\mathtt{n+1}}, \rho) : \mathtt{Nat}[\mathrm{M}, \mathrm{N}]$, where

$$\mathrm{I}_{(\underline{\mathtt{n+1}},\rho)} \equiv \mathrm{I}_{\underline{\mathtt{n}}} + \mathrm{K}_1 + \ldots + \mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{K}_n} \mathrm{I}_{c_n}.$$

By (4.14), the thesis can be easily reached.

- Consider the case $D = (\mathtt{fix}\ x.u, \rho, \xi)$. Yet another time, we want to prove Point 1, that is $E = (u, (\mathtt{fix}\ x.u, \rho) \cdot \rho, \xi)$ is such that $\phi; \Phi \vdash_{\mathrm{J}} E : \sigma$, where $\phi; \Phi \models \mathrm{I} = \mathrm{J}$ and $|D| > |E|$. The latter is easy, as usual:

$$|D| = |\mathtt{fix}\ x.u| + |\xi| > |u| + |\xi| = |E|,$$

so we consider the former. By inspecting a proof of $\phi; \Phi \vdash^{\mathcal{E}}_{\mathrm{I}} D : \sigma$, we can easily derive the following judgments (where $\rho \equiv c_1, \ldots, c_n$):

$$\phi, b; \Phi, b < \mathrm{H}; x_1 : [a < \mathrm{K}_1] \cdot \mu_1, \ldots, x_n : [a < \mathrm{K}_n] \cdot \mu_n, x : [a < \mathrm{L}] \cdot \gamma \vdash_{\mathrm{I}_u} u : \tau; \tag{4.17}$$

$$\phi, a; \Phi, a < \mathrm{M}_i \vdash_{\mathrm{I}_{c_i}} c_i : \eta_i; \tag{4.18}$$

$$\phi; \Phi \vdash_{\mathrm{I}_\xi} \xi : (\delta, \sigma). \tag{4.19}$$

Moreover:

$$\begin{aligned}
&\phi; \Phi \vdash \tau\{\mathbf{0}/b\} \sqsubseteq \delta;\\
&\phi, a, b; \Phi, a < \mathrm{L}, b < \mathrm{H} \vdash \tau\{\textstyle\bigcirc\!\!\!\!\!\!\triangle\limits_{b}^{b+\mathbf{1},a} \mathrm{L} + b + \mathbf{1}/b\} \sqsubseteq \gamma;\\
&\phi; \Phi \vdash [a < \mathrm{M}_i] \cdot \eta_i \sqsubseteq \sum_{b<\mathrm{H}} [a < \mathrm{K}_i] \cdot \mu_i;\\
&\phi; \Phi \models \textstyle\bigcirc\!\!\!\!\!\!\triangle\limits_{b}^{\mathbf{0},\mathbf{1}} \mathrm{L} \leqslant \mathrm{H}, \mathrm{N};\\
&\phi; \Phi \models \mathrm{I} \geqslant \mathrm{N} \dot{-} \mathbf{1} + \sum_{b<\mathrm{H}} \mathrm{I}_u + \mathrm{M}_1 + \ldots + \mathrm{M}_n + \sum_{a<\mathrm{M}_1} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{M}_n} \mathrm{I}_{c_n} + \mathrm{I}_\xi.
\end{aligned}$$

By manipulations of the indices similar to the one used in the proof of Subject Reduction, we can derive the following from (4.17), given the judgments above:

$$\begin{aligned}
&\phi; \Phi; \Gamma, x : [a < \mathrm{L}\{\mathbf{0}/b\}] \cdot \gamma\{\mathbf{0}/b\} \vdash_{\mathrm{I}_u\{\mathbf{0}/b\}} u : \delta;\\
&\phi; \Phi, a < \mathrm{L}\{\mathbf{0}/b\}; \Delta \vdash_{\mathrm{P}\{a/c\} \dot{-} \mathbf{1} + \sum_{b<\mathrm{P}\{a/c\}} \mathrm{I}_u\{\mathrm{R}/b\}} \mathtt{fix}\ x.u : \gamma\{\mathbf{0}/b\}.
\end{aligned}$$

In the equations above,

$$\mathrm{P} \equiv \textstyle\bigcirc\!\!\!\!\!\!\triangle\limits_{b}^{\mathbf{0},\mathbf{1}} \mathrm{L}\{b + \mathbf{1} + \bigcirc\!\!\!\!\!\!\triangle\limits_{b}^{\mathbf{0},c} \mathrm{L}/b\};$$

$$\mathrm{R} \equiv \mathbf{1} + b + \sum_{c<a} \mathrm{P};$$

and Γ, Δ can be chosen in such a way as to guarantee:

$$\begin{aligned}
\phi; \Phi \vdash x_1 : [a < \mathrm{M}_1] \cdot \eta_1, \ldots, x_n : [a < \mathrm{M}_n] \cdot \eta_n &\cong \sum_{a<\mathrm{L}\{\mathbf{0}/b\}} \Delta \uplus \Gamma\\
&\sqsubseteq x_1 : \sum_{b<\mathrm{H}} [a < \mathrm{K}_1] \cdot \mu_1, \ldots, x_n : \sum_{b<\mathrm{H}} [a < \mathrm{K}_n] \cdot \mu_n.
\end{aligned}$$

So we have that

$$\phi;\Phi \vdash_{\mathrm{I}_{(u,(\mathtt{fix}\ x.u,\rho)\cdot\rho)}} (u,(\mathtt{fix}\ x.u,\rho)\cdot\rho) : \delta,$$

where

$$\mathrm{I}_{(u,(\mathtt{fix}\ x.u,\rho)\cdot\rho)} \equiv \mathrm{I}_u\{\mathbf{0}/b\} + \mathrm{L}\{\mathbf{0}/b\} + \sum_{a<\mathrm{L}\{\mathbf{0}/b\}} (\mathrm{P}\{a/c\} \dot{-} \mathbf{1} + \sum_{b<\mathrm{P}\{a/c\}} \mathrm{I}_u\{\mathrm{N}/b\}) + \mathrm{M}_1 + \ldots + \mathrm{M}_n + \sum_{a<\mathrm{M}_1} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{M}_n} \mathrm{I}_{c_n}.$$

The value of $\mathrm{I}_{(u,(\mathtt{fix}\ x.u,\rho)\cdot\rho)}$ can then be proved to be equal or smaller than

$$\mathrm{N} \dot{-} \mathbf{1} + \sum_{b<\mathrm{H}} \mathrm{I}_u + \mathrm{M}_1 + \ldots + \mathrm{M}_n + \sum_{a<\mathrm{M}_1} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{M}_n} \mathrm{I}_{c_n},$$

under the hypotheses in ϕ. This immediately yields the thesis, given (4.19).

- Consider the case $D = (x_m, ((t_0,\rho_0),\ldots,(t_n,\rho_n)),\xi)$. We want to prove Point 2, that is $E = (t_m,\rho_m,\xi)$ is such that $\phi;\Phi \vdash_{\mathrm{J}} E : \sigma$, where $\phi;\Phi \models \mathrm{I} > \mathrm{J}$ and $|E| < |D| + |t|$. The latter is immediate by Lemma 4.2, so we consider the former. By inspecting a proof of $\phi;\Phi \vdash^{\mathcal{E}}_{\mathrm{I}} D : \sigma$, we can easily derive the following judgments

$$\phi;\Phi;x_1 : [a<\mathrm{K}_1]\cdot\mu_1,\ldots,x_n : [a<\mathrm{K}_n]\cdot\mu_n \vdash_{\mathrm{I}_{x_m}} x_m : \tau; \tag{4.20}$$

$$\phi,a;\Phi,a<\mathrm{K}_i \vdash_{\mathrm{I}_{(t_i,\rho_i)}} (t_i,\rho_i) : \mu_i; \tag{4.21}$$

$$\phi;\Phi \vdash_{\mathrm{I}_\xi} \xi : (\tau,\sigma). \tag{4.22}$$

Moreover:

$$\phi;\Phi \models \mathrm{K}_m \geqslant \mathbf{1}; \tag{4.23}$$

$$\phi;\Phi \vdash \mu_m\{\mathbf{0}/a\} \sqsubseteq \tau; \tag{4.24}$$

$$\phi;\Phi \vdash \mathrm{I} \geqslant \mathrm{I}_{x_m} + \mathrm{K}_1 + \ldots + \mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{I}_{(t_1,\rho_1)} + \ldots + \sum_{a<\mathrm{K}_n} \mathrm{I}_{(t_n,\rho_n)} + \mathrm{I}_\xi. \tag{4.25}$$

From (4.21) where $i = m$, (4.23), and (4.24), one obtains that $\phi;\Phi \vdash_{\mathrm{I}_{(t_m,\rho_m)}\{\mathbf{0}/a\}} (t_m,\rho_m) : \tau$ and, by (4.22), that

$$\phi;\Phi \vdash_{\mathrm{I}_{(t_m,\rho_m)}\{\mathbf{0}/a\}+\mathrm{I}_\xi} E : \sigma.$$

But from (4.25) and (4.23) one easily infer that

$$\phi;\Phi \models \mathrm{I} > \mathrm{I}_{(t_m,\rho_m)}\{\mathbf{0}/a\} + \mathrm{I}_\xi,$$

that is the thesis.

This concludes the proof. □

It is worth noticing that if Φ is inconsistent, the inequality $\phi;\Phi \models \mathrm{I} > \mathrm{J}$ in Lemma 4.5, Point 2, does not necessary imply that weight strictly decreases. Indeed, Intensional Soundness only holds in presence of a consistent set of constraints:

Theorem 4.6 (Intensional Soundness)**.** *Let* $\vdash_{\mathrm{I}} t : \mathtt{Nat}[\mathrm{J},\mathrm{K}]$ *and* $t \Downarrow^n \underline{\mathtt{m}}$*. Then,* $n \leqslant |t|\cdot([\![\mathrm{I}]\!]+1)$.

Proof. By induction on n, making essential use of Lemma 4.5 and Lemma 4.2. □

Please observe that an easy consequence of Theorem 4.6 is intensional soundness *for functions*. As an example, if $a;\varnothing;\varnothing \vdash_{\mathrm{I}} t : [b < \mathrm{J}]\cdot\mathtt{Nat}[a] \multimap \mathtt{Nat}[\mathrm{K},\mathrm{H}]$, then the complexity of evaluating $t\ \underline{\mathtt{n}}$ is at most $(|t\ \underline{\mathtt{n}}|)\cdot([\![\mathrm{I}\{\mathbf{n}/a\}]\!]+1)$. Observe, however, that $|t\ \underline{\mathtt{n}}|$ does not depend on n, since $|\underline{\mathtt{n}}| = 1$.

5. Relative Completeness

This section is devoted to proving *relative completeness* for the type system $\mathsf{d}\ell\mathsf{PCF}$. In fact, *two* relative completeness theorems will be presented. The first one (Theorem 5.6) states relative completeness *for programs*: for each PCF program t that evaluates to a numeral $\underline{\mathtt{n}}$ there is a type derivation in $\mathsf{d}\ell\mathsf{PCF}$ whose index terms capture both the number of reduction steps and the value of $\underline{\mathtt{n}}$. The second one (Theorem 5.12) states relative completeness *for functions*: for each PCF term $t : \mathtt{Nat} \to \mathtt{Nat}$ computing a *total* function f in time expressed by a function g there exists a type derivation in $\mathsf{d}\ell\mathsf{PCF}$ whose index terms capture both the extensional behavior f and the intensional property embedded into g.

Relative completeness does not hold in general. Indeed, it holds only when the underlying equational program $\mathcal{E}$ is *universal*, i.e. when it is sufficiently expressive as to encode all total computable functions. A universal equational program is introduced in Section 5.1.

Relative completeness for programs will be proved using a weighted form of *Subject Expansion* (Theorem 5.5) similar to the one holding in intersection type theories. This will be proved in Section 5.2. The proof of relative completeness for functions needs a further step: a *uniformization* result (Lemma 5.11) relying on the properties of the universal model. This is the subject of Section 5.3.

5.1. Universal Equational Program.

Since the class of equational programs is clearly recursively enumerable, it can be put in one-to-one correspondence with natural numbers, using a coding scheme $\ulcorner\cdot\urcorner$ *à la Gödel*. Such a coding, as usual, can be used to define a *universal equational program* $\mathcal{U}$ that is able to simulate all equational programs (including itself).

Let $\ulcorner\mathcal{E},\mathtt{f}\urcorner$ be the natural number coding an equational program $\mathcal{E}$ and a function symbol $\mathtt{f}$ among the ones defined in it. This can be easily computed from (a description of) $\mathcal{E}$ and $\mathtt{f}$. A signature $\Sigma_{\mathcal{U}}$ containing just the symbol $\mathtt{empty}$ of arity 0 and the symbols $\mathtt{pair}$ and $\mathtt{eval}$ of arity 2 (plus some auxiliary symbols) is sufficient to define the universal program $\mathcal{U}$. For each $\mathtt{f}$ of arity n, the equational program $\mathcal{U}$ satisfies

$$[\![\mathtt{eval}(\ulcorner\mathcal{E},\mathtt{f}\urcorner, \mathit{pairing}_n(x_1,\ldots,x_n))]\!]^{\mathcal{U}}_{\rho} = [\![\mathtt{f}(x_1,\ldots,x_n)]\!]^{\mathcal{E}}_{\rho},$$

where $\mathit{pairing}_n(t_1,\ldots,t_n)$ is defined by induction on n:

$$\mathit{pairing}_0 \equiv \mathtt{empty};$$
$$\mathit{pairing}_{n+1}(t_1,\ldots,t_{n+1}) \equiv \mathtt{pair}(\mathit{pairing}_n(t_1,\ldots,t_n), t_{n+1}).$$

This way, $\mathcal{U}$ acts as an interpreter for any equational program. Such a universal program $\mathcal{U}$ can be defined as a *finite* sequence of equations, similarly to what happens in the construction of, e.g., universal Turing machines.

The universal equational program $\mathcal{U}$ enjoys some nice properties which are crucial when proving Subject Expansion. The following lemma says, for example, that sums and bounded sums can always be formed (modulo $\cong$) whenever index terms are built and reasoned about using the universal program:

Lemma 5.1. *1. For every A and B such that $\phi;\Phi \vdash^{\mathcal{U}} A \Downarrow$, $\phi;\Phi \vdash^{\mathcal{U}} B \Downarrow$, and $(\!|A|\!) = (\!|B|\!)$, there are C and D such that $\phi;\Phi \vdash^{\mathcal{U}} C \cong A$, $\phi;\Phi \vdash^{\mathcal{U}} D \cong B$ and $C \uplus D$ is defined.*
2. For every A and I such that $\phi, a;\Phi, a < \mathrm{I} \vdash^{\mathcal{U}} A \Downarrow$ and $\phi;\Phi \vdash^{\mathcal{U}} \mathrm{I} \Downarrow$, there is B such that $\phi, a;\Phi, a < \mathrm{I} \vdash^{\mathcal{U}} B \cong A$ and $\sum_{a<\mathrm{I}} B$ is defined.

Proof. These are inductions on the structure of the involved formulas. Actually, it is convenient to enrich the statements above (which only deals with *modal* types) with similar statements involving *basic* types, this way facilitating the inductive argument. □

5.2. **Subject Expansion and Relative Completeness for Programs.** *Weighted Subject Expansion* (Theorem 5.5 below) says that typing is preserved while weights increase by at most one along any $\mathsf{K_{PCF}}$ expansion step. This is somehow the converse of Weighted Subject Reduction. Weighted Subject Expansion, however, does not hold in general but only when the underlying equational program is universal.

In order to prove Weighted Subject Expansion, only typing that carry precise information should be considered. As an example, we write $\phi;\Phi \Vdash_{\mathrm{I}} C : \sigma$ if we can derive $\phi;\Phi \vdash_{\mathrm{I}} C : \sigma$ by *precise* type derivations. The type of a precisely-typable configuration, in other words, carries exact information about the value of the objects at hand. One can easily extend the above notation to type derivations for closures and stacks. Recall that a precise type derivation is a type derivation such that all premises in the form $\sigma \sqsubseteq \tau$ (respectively, in the form $\mathrm{I} \leqslant \mathrm{J}$) are actually required to be in the form $\sigma \cong \tau$ (respectively, $\mathrm{I} = \mathrm{J}$).

Furthermore, only specific typing transformations should be considered, namely those that leave the weight information unaltered. In order to achieve this, some properties of precise typability for the $\mathsf{K_{PCF}}$ machine should be exploited. As an example, if a closure $\phi;\Phi \Vdash_{\mathrm{I}} (t,\rho) : \sigma$, then $\phi;\Phi \Vdash_{\mathrm{J}} (t,\rho) : \tau$ whenever τ and J such that $\phi;\Phi \vdash \sigma \cong \tau$ and $\phi;\Phi \models \mathrm{I} = \mathrm{J}$. This is a natural variation on the Subtyping Lemma for terms (Lemma 3.7).

Finally, it is worth noticing that by considering an inconsistent set of constraints Φ, it is possible to make any closure (t,ρ) typable with type σ (in the sense of PCF) to be also typable in the sense of $\mathsf{d}\ell\mathsf{PCF}$: $\phi;\Phi \Vdash_{\mathrm{I}} (t,\rho) : \tau$ whenever $(\!|\tau|\!) = \sigma$ and for every index term I. This says that inconsistent sets cover a role similar to the ω-rule in intersection type systems.

The following two lemmas will be useful in the sequel, and allow to "join" apparently uncorrelated typing judgements into one:

Lemma 5.2. *Let θ be the substitution $\{a + \mathrm{I}/a\}$. Suppose that $\pi \rhd \phi, a;\Phi, a < \mathrm{I} \Vdash_{\mathrm{H}} c : \sigma$, that $\rho \rhd \phi, a;\Phi\theta, a < \mathrm{J} \Vdash_{\mathrm{H}\theta} c : \sigma\theta$, and that $(\!|\pi|\!) = (\!|\rho|\!)$. Then, $\phi, a;\Phi, a < \mathrm{I} + \mathrm{J} \Vdash_{\mathrm{H}} c : \sigma$.*

Proof. By simultaneous induction on π and ρ. We make essential use of the implicit assumption about the universality of the underlying equational program. □

Lemma 5.3. *Let θ be the substitution $\{\sum_{c<a} \mathrm{J}\{c/a\} + b/c\}$. Suppose that $\pi \rhd \phi, a, b;\Phi\theta, a < \mathrm{I}, b < \mathrm{J} \Vdash_{\mathrm{H}\theta} c : \sigma\theta$. Then, $\phi, a;\Phi, c < \sum_{a<\mathrm{I}} \mathrm{J} \Vdash_{\mathrm{H}} c : \sigma$.*

Proof. By induction on the derivation π, again using the properties of a universal equational program. □

But there are even other ways to turn two typing derivations into a more general one, again relying on the semantic nature of $\mathsf{d}\ell\mathsf{PCF}$:

Lemma 5.4. *Suppose that* $\pi \rhd \phi; \Phi, \mathrm{I} \leqslant \mathrm{J} \Vdash_{\mathrm{K}} c : \sigma$*, that* $\rho \rhd \phi; \Phi, \mathrm{I} > \mathrm{J} \Vdash_{\mathrm{K}} c : \sigma$*, and that* $(\!|\pi|\!) = (\!|\rho|\!)$*. Then,* $\phi; \Phi \Vdash_{\mathrm{K}} c : \sigma$*.*

It is now time to state Weighted Subject Expansion, since all the necessary ingredients have been introduced:

Theorem 5.5 (Weighted Subject Expansion)**.** *Suppose that* $\pi \rhd \phi; \Phi \Vdash_{\mathrm{I}} D : \sigma$ *and that* $\rho \to (\!|\pi|\!)$*, where* $\rho \rhd\!\vdash C : (\!|\sigma|\!)$*. Then* $\nu \rhd \phi; \Phi \Vdash_{\mathrm{J}} C : \sigma$*, where* $\phi; \Phi \models \mathrm{J} \leqslant \mathrm{I} + \boldsymbol{1}$ *and* $(\!|\nu|\!) = \rho$*. Moreover,* ν *can be effectively computed from* π *and* ρ*.*

Proof. The proof is by cases on the shape of the reduction $C \to D$. We just present some cases, the others can be obtained analogously.

- Consider the case
$$C \equiv (\underline{0}, \rho, (t, u, \mu) \cdot \xi) \to (t, \mu, \xi) \equiv D.$$
By assumption we have that C is typable in PCF and that $\phi; \Phi \Vdash_{\mathrm{I}} D : \sigma$. So, we have that
$$\phi; \Phi \Vdash_{\mathrm{I}_{(t,\mu)}} (t, \mu) : \tau;$$
$$\phi; \Phi \Vdash_{\mathrm{I}_{\xi}} \xi : (\tau, \sigma);$$
$$\phi; \Phi \models \mathrm{I} = \mathrm{I}_{(t,\mu)} + \mathrm{I}_{\xi};$$
for some $\mathrm{I}_{(t,\mu)}$ and I_{ξ}. We clearly also have that $\phi; \Phi, \mathbf{0} \leqslant \mathbf{0} \Vdash_{\mathrm{I}_{(t,\mu)}} (t, \mu) : \tau$. $\Phi, \mathbf{1} \leqslant \mathbf{0}$ is an inconsistent set of constraints, and since C is typable in PCF (as remarked above), we also have that $\phi; \Phi, \mathbf{1} \leqslant \mathbf{0} \Vdash_{\mathrm{I}_{(t,\mu)}} (u, \mu) : \tau$. This implies, in particular, that $\phi, \Phi \Vdash_{\mathrm{I}} (t, u, \mu) \cdot \xi : (\mathtt{Nat}[\mathbf{0}], \sigma)$. Now, assume that $\rho = (t_1, \rho_1) \cdot \ldots \cdot (t_n, \rho_n)$ where for every $1 \leqslant i \leqslant n$, (t_i, ρ_i) is typable in PCF. Since $\Phi, a < \mathbf{0}$ is inconsistent, we have that
$$\phi, a; \Phi, a < \mathbf{0} \Vdash_{\mathbf{0}} (t_i, \rho_i) : \mu_i$$
for some μ_i. By Lemma 3.8 we can build a derivation for
$$\phi; \Phi; x_1 : [a < \mathbf{0}] \cdot \mu_1, \ldots, x_n : [a < \mathbf{0}] \cdot \mu_n \Vdash_{\mathbf{0}} \underline{0} : \mathtt{Nat}[\mathbf{0}].$$
So, we have that
$$\phi; \Phi \Vdash_{\mathbf{0}} (\underline{0}, \rho) : \mathtt{Nat}[\mathbf{0}].$$
Summing up, we obtain that
$$\phi; \Phi \Vdash_{\mathrm{I}} C : \sigma,$$
from which the thesis easily follows, since $\phi; \Phi \models \mathrm{I} \leqslant \mathrm{I} + \mathbf{1}$.
- Consider the case
$$C \equiv (\lambda x.t, \rho, c \cdot \xi) \to (t, c \cdot \rho, \xi) \equiv D.$$
By assumption we have that C is typable in PCF and that $\phi; \Phi \Vdash_{\mathrm{I}} D : \sigma$. So, we have that
$$\phi; \Phi; x_1 : [a < \mathrm{K}_1] \cdot \tau_1, \ldots, x_n : [a < \mathrm{K}_n] \cdot \tau_n \Vdash_{\mathrm{I}_t} t : \mu;$$
$$\phi, a; \Phi, a < \mathrm{K}_i \Vdash_{\mathrm{I}_{c_i}} c_i : \tau_i;$$
$$\phi; \Phi \Vdash_{\mathrm{I}_{\xi}} \xi : (\mu, \sigma);$$

where:

$$\phi;\Phi \models \mathrm{I} = \mathrm{I}_t + \mathrm{K}_1 + \ldots + \mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{K}_n} \mathrm{I}_{c_n} + \mathrm{I}_\xi.$$

For simplicity and without loosing any generality, we can consider the case where $c \cdot \rho \equiv c_1 \ldots c_n$ with $x \equiv x_1$ and $c \equiv c_1$. So, in particular we can build a derivation ending as follows:

$$\frac{\phi;\Phi;x_1 : [a < \mathrm{K}_1] \cdot \tau_1, \ldots, x_n : [a < \mathrm{K}_n] \cdot \tau_n \Vdash_{\mathrm{I}_t} t : \mu}{\phi;\Phi;x_2 : [a < \mathrm{K}_2] \cdot \tau_2, \ldots, x_n : [a < \mathrm{K}_n] \cdot \tau_n \Vdash_{\mathrm{I}_t} \lambda x_1.t : [a < \mathrm{K}_1] \cdot \tau_1 \multimap \mu}$$

and thus we have that $\phi;\Phi \Vdash_{\mathrm{I}_{(\lambda x.t,\rho)}} (\lambda x.t,\rho) : [a < \mathrm{K}_1] \cdot \tau_1 \multimap \mu$, where

$$\mathrm{I}_{(\lambda x.t,\rho)} \equiv \mathrm{I}_t + \mathrm{K}_2 + \ldots + \mathrm{K}_n + \sum_{a<\mathrm{K}_2} \mathrm{I}_{c_2} + \ldots + \sum_{a<\mathrm{K}_n} \mathrm{I}_{c_n}.$$

Further, we have that

$$\phi;\Phi \Vdash_{\mathrm{I}_\xi + \mathrm{K}_1 + \sum_{a<\mathrm{K}_1} \mathrm{I}_{c_1}} c_1 \cdot \xi : ([a < \mathrm{K}_1] \cdot \tau_1 \multimap \mu, \sigma)$$

and, as an easy consequence, that

$$\phi;\Phi \Vdash_{\mathrm{I}_{(\lambda x.t,\rho)} + \mathrm{I}_\xi + \mathrm{K}_1 + \sum_{a<\mathrm{K}_1} \mathrm{I}_{c_1}} C : \sigma.$$

This easily leads to the conclusion, since

$$\begin{aligned}\phi;\Phi \models \mathrm{I} &= \mathrm{I}_t + \mathrm{K}_1 + \ldots + \mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{K}_n} \mathrm{I}_{c_n} + \mathrm{I}_\xi \\ &= \mathrm{I}_{(\lambda x.t,\rho)} + \mathrm{I}_\xi + \mathrm{K}_1 + \sum_{a<\mathrm{K}_1} \mathrm{I}_{c_1}.\end{aligned}$$

- Consider the case

$$C \equiv (\mathtt{fix}\ x.t, \rho, \xi) \to (t, (\mathtt{fix}\ x.t, \rho) \cdot \rho, \xi) \equiv D.$$

By assumption we have that C is typable in PCF and that $\phi;\Phi \Vdash_{\mathrm{I}} D : \sigma$. So, we have that

$$\phi;\Phi;x_1 : [a < \mathrm{K}_1] \cdot \tau_1, \ldots, x_n : [a < \mathrm{K}_n] \cdot \tau_n \Vdash_{\mathrm{I}_t} t : \mu; \tag{5.1}$$

$$\phi, a;\Phi, a < \mathrm{K}_i \Vdash_{\mathrm{I}_{c_i}} c_i : \tau_i; \tag{5.2}$$

$$\phi;\Phi \Vdash_{\mathrm{I}_\xi} \xi : (\mu, \sigma); \tag{5.3}$$

where:

$$\phi;\Phi \models \mathrm{I} = \mathrm{I}_t + \mathrm{K}_1 + \ldots + \mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{K}_n} \mathrm{I}_{c_n} + \mathrm{I}_\xi. \tag{5.4}$$

For simplicity and without losing any generality, we can consider the case where $(\mathtt{fix}\ x.t, \rho) \cdot \rho \equiv c_1 \ldots c_n$ with $x \equiv x_1$ and $(\mathtt{fix}\ x.t, \rho) \equiv c_1$. As a consequence, we can conclude that:

$$\phi, a;\Phi, a < \mathrm{K}_1;\Gamma \Vdash_{\mathrm{I}_{\mathtt{fix}\ x.t}} \mathtt{fix}\ x.t : \tau_1; \tag{5.5}$$

$$\phi, a, b;\Phi, a < \mathrm{K}_1, b < \mathrm{H}_i \Vdash_{\mathrm{J}_{c_i}} c_i : \mu_i; \tag{5.6}$$

where $\Gamma \equiv x_2 : [b < \mathrm{H}_2] \cdot \mu_2, \ldots, x_n : [b < \mathrm{H}_n] \cdot \mu_n$, and

$$\phi, a;\Phi, a < \mathrm{K}_1 \models \mathrm{I}_{c_1} = \mathrm{I}_{\mathtt{fix}\ x.t} + \mathrm{H}_2 + \ldots + \mathrm{H}_n + \sum_{b<\mathrm{H}_2} \mathrm{J}_{c_2} + \ldots + \sum_{b<\mathrm{H}_n} \mathrm{J}_{c_n}. \tag{5.7}$$

Our objective now is to prove that

$$\phi, \Phi \Vdash_{\mathrm{I}_{(\mathtt{fix}\ x.t,\rho)}} (\mathtt{fix}\ x.t, \rho) : \mu, \tag{5.8}$$

where $\phi, \Phi \models \mathrm{I}_{(\mathtt{fix}\ x.t,\rho)} = \mathrm{I} \doteq \mathrm{I}_\xi$. The thesis easily follows from (5.8). To do that, we proceed by spelling out what the premises of (5.5) are. They are:

$$\phi, a, b; \Phi, a < \mathrm{K}_1, b < \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{P}; x : [c < \mathrm{P}] \cdot \gamma\{ \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{b+1,c} \mathrm{P} + b + \mathbf{1}/b\}, \Delta \Vdash_{\mathrm{J}_t} t : \gamma, \tag{5.9}$$

and the following two:

$$\phi, a; \Phi, a < \mathrm{K}_1 \Vdash \tau_1 \cong \gamma\{\mathbf{0}/b\};$$
$$\phi, a; \Phi, a < \mathrm{K}_1 \Vdash \Gamma \cong \sum_{b < \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{P}} \Delta;$$

where P and J_t are index terms such that

$$\phi, a; \Phi, a < \mathrm{K}_1 \models \mathrm{I}_{\mathtt{fix}\ x.t} = \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{P} \doteq \mathbf{1} + \sum_{b < \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{P}} \mathrm{J}_t. \tag{5.10}$$

Now, consider an index term N such that

$$\phi; \Phi \models \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{N} = \mathbf{1} + \sum_{a < \mathrm{K}_1} \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{P}$$

Such an index term can be easily defined from P and K_1, given that the underlying equational program is assumed to be universal. For the same reasons, one can define types δ and η, a type context Σ and an index term R such that the following holds (where θ is $\{\mathbf{1} + \sum_{a<a} \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{P} + b/b\}$):

$$\phi; \Phi \Vdash \eta\{\mathbf{0}/b\} = \mu; \qquad \phi, a, b; \Phi, a < \mathrm{K}_1, b < \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{P} \Vdash \eta\theta = \gamma;$$

$$\phi; \Phi \Vdash \delta\{\mathbf{0}/b\} = \tau_1; \qquad \phi, a, b, c; \Phi, a < \mathrm{K}_1, b < \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{P}, c < \mathrm{P}, \Vdash \delta\theta = \gamma\{ \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{b+1,c} \mathrm{P} + b + \mathbf{1}/b\};$$

$$\phi; \Phi \Vdash \mathrm{R}\{\mathbf{0}/b\} = \mathrm{I}_t; \qquad \phi, a, b; \Phi, a < \mathrm{K}_1, b < \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{P} \models \mathrm{R}\theta = \mathrm{J}_t;$$

$$\phi; \Phi \Vdash \Sigma\{\mathbf{0}/b\} \cong \Gamma; \qquad \phi, a, b; \Phi, a < \mathrm{K}_1, b < \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{P} \Vdash \Sigma\theta \cong \Delta.$$

This is possible since the type derivations for (5.1) and (5.9) have exactly the same PCF skeleton. By transforming them according to the equations above, one can merge them into one with conclusion:

$$\phi, b; \Phi, b < \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{N}; x : [a < \mathrm{N}] \cdot \delta, \Sigma \Vdash_{\mathrm{R}} t : \eta.$$

So, by using again the R rule we obtain:

$$\phi; \Phi; \sum_{b < \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{N}} \Sigma \Vdash_{\bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{N} \doteq \mathbf{1} + \sum_{b < \bigcirc\!\!\!\!\!\!\triangle_{\ b}^{\mathbf{0},\mathbf{1}} \mathrm{N}} \mathrm{R}} \mathtt{fix}\ x.t : \mu.$$

We are not at (5.8), however: it is still necessary to type ρ appropriately. But note that we have:

$$\phi, \Phi \Vdash \sum_{b<\bigtriangleup_b^{\mathbf{0,1}} \mathrm{N}} \Sigma = \Gamma \uplus \sum_{b<\bigtriangleup_b^{\mathbf{0,1}} \mathrm{N} \dot{-} 1} \Delta = \Gamma \uplus \sum_{a<\mathrm{K}_1} \sum_{b<\bigtriangleup_b^{\mathbf{0,1}} \mathrm{P}} \Delta = \Gamma \uplus \sum_{a<\mathrm{K}_1} \Gamma.$$

So we can find types $\beta_2, \ldots, \beta_n$ such that

$$\sum_{b<\bigtriangleup_b^{\mathbf{0,1}} \mathrm{N}} \Sigma = x_2 : [a < \mathrm{K}_2 + \sum_{a<\mathrm{K}_1} \mathrm{H}_2] \cdot \beta_2, \ldots, x_n : [a < \mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{H}_n] \cdot \beta_n,$$

where for every $2 \leqslant i \leqslant n$,

$$\phi, a; \Phi, a < \mathrm{K}_i \Vdash \beta_i \cong \tau_i;$$
$$\phi, a; \Phi, a < \mathrm{K}_1, b < \mathrm{H}_i \Vdash \beta_i\{\mathrm{K}_i + b + \sum_{a<a} \mathrm{H}_i/a\} \cong \mu_i.$$

Similarly, one can define index terms $\mathrm{Q}_2, \ldots, \mathrm{Q}_n$ such that

$$\phi, a; \Phi, a < \mathrm{K}_i \models \mathrm{Q}_i = \mathrm{I}_{c_i};$$
$$\phi, a; \Phi, a < \mathrm{K}_1, b < \mathrm{H}_i \models \mathrm{Q}_i\{\mathrm{K}_i + b + \sum_{a<a} \mathrm{H}_i/a\} = \mathrm{J}_{c_i}.$$

By relabelling the type derivations of (5.2) and (5.6) (which are structurally equal) according to the types and index terms introduced above, one obtains:

$$\phi, a; \Phi, a < \mathrm{K}_1 + \sum_{a<\mathrm{K}_1} \mathrm{H}_i \Vdash_{\mathrm{Q}_i} c_i : \beta_i;$$

From this it follows that $\phi; \Phi \Vdash_{\mathrm{I}_{(\mathtt{fix}\ x.t,\rho)}} (\mathtt{fix}\ x.t, \rho) : \mu$, where

$$\mathrm{I}_{(\mathtt{fix}\ x.t,\rho)} \equiv \Big(\bigcirc\!\!\!\!\bigtriangleup_b^{\mathbf{0,1}} \mathrm{N} \dot{-} \mathbf{1} + \sum_{b<\bigtriangleup_b^{\mathbf{0,1}} \mathrm{N}} \mathrm{R} \Big) + \Big(\mathrm{K}_2 + \sum_{a<\mathrm{K}_1} \mathrm{H}_2 + \cdots + \mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{H}_n + \sum_{a<(\mathrm{K}_2 + \sum_{a<\mathrm{K}_1} \mathrm{H}_2)} \mathrm{Q}_2 + \cdots + \sum_{a<(\mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{H}_n)} \mathrm{Q}_n \Big).$$

Let us separately analyze the two thunks in which the expression above can be decomposed. On the one hand we have that:

$$\phi; \Phi \models \bigcirc\!\!\!\!\bigtriangleup_b^{\mathbf{0,1}} \mathrm{N} \dot{-} \mathbf{1} + \sum_{b<\bigtriangleup_b^{\mathbf{0,1}} \mathrm{N}} \mathrm{R} = \sum_{a<\mathrm{K}_1} \bigcirc\!\!\!\!\bigtriangleup_b^{\mathbf{0,1}} \mathrm{P} + \mathrm{I}_t + \sum_{a<\mathrm{K}_1} \sum_{b<\bigtriangleup_b^{\mathbf{0,1}} \mathrm{P}} \mathrm{J}_t$$
$$= \sum_{a<\mathrm{K}_1} \mathrm{I}_{\mathtt{fix}\ x.t} + \mathrm{K}_1 + \mathrm{I}_t.$$

On the other hand, let us observe that

$$\phi; \Phi \models \sum_{a<(\mathrm{K}_2 + \sum_{a<\mathrm{K}_1} \mathrm{H}_2)} \mathrm{Q}_2 + \cdots + \sum_{a<(\mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{H}_n)} \mathrm{Q}_n$$
$$= \sum_{a<\mathrm{K}_2} \mathrm{I}_{c_2} + \sum_{a<\mathrm{K}_2} \sum_{b<\mathrm{H}_2} \mathrm{J}_{c_2} + \ldots + \sum_{a<\mathrm{K}_n} \mathrm{I}_{c_n} + \sum_{a<\mathrm{K}_n} \sum_{b<\mathrm{H}_n} \mathrm{J}_{c_n}.$$

Combining the equations above with (5.4), (5.7) and (5.10), one easily reaches $\phi; \Phi \models \mathrm{I}_{(\mathtt{fix}\ x.t,\rho)} = \mathrm{I} \dot{-} \mathrm{I}_\xi$, which is the thesis.

- Consider the case

$$C \equiv (\ \texttt{ifz}\ w\ \texttt{then}\ u\ \texttt{else}\ v, \rho, \xi) \to (w, \rho, (u, v, \rho) \cdot \xi) \equiv D.$$

By assumption we have that C is typable in PCF and that $\phi; \Phi \Vdash_{\mathrm{I}} D : \sigma$. So, we have that

$$\phi; \Phi; x_1 : [a < \mathrm{K}_1] \cdot \tau_1, \ldots, x_n : [a < \mathrm{K}_n] \cdot \tau_n \Vdash_{\mathrm{I}_w} w : \texttt{Nat}[\mathrm{H}]; \tag{5.11}$$

$$\phi, a; \Phi, a < \mathrm{K}_i \Vdash_{\mathrm{I}_{c_i}} c_i : \tau_i; \tag{5.12}$$

$$\phi; \Phi, \mathrm{H} \leqslant \mathbf{0} \Vdash_{\mathrm{I}_{(u,v,\rho)}} (u, \rho) : \mu; \tag{5.13}$$

$$\phi; \Phi, \mathbf{1} \leqslant \mathrm{H} \Vdash_{\mathrm{I}_{(u,v,\rho)}} (v, \rho) : \mu; \tag{5.14}$$

$$\phi; \Phi \Vdash_{\mathrm{I}_\xi} \xi : (\mu, \sigma); \tag{5.15}$$

where $\rho \equiv c_1 \ldots c_n$. Moreover:

$$\phi; \Phi \models \mathrm{I} = \mathrm{I}_w + \mathrm{K}_1 + \ldots + \mathrm{K}_n + \sum_{a<\mathrm{K}_1} \mathrm{I}_{c_1} + \ldots + \sum_{a<\mathrm{K}_n} \mathrm{I}_{c_n} + \mathrm{I}_{(u,v,\rho)} + \mathrm{I}_\xi. \tag{5.16}$$

By further spelling out (5.13) and (5.14), we obtain the following:

$$\phi; \Phi, \mathrm{H} \leqslant \mathbf{0}; x_1 : [a < \mathrm{H}_1] \cdot \gamma_1, \ldots, x_n : [a < \mathrm{H}_n] \cdot \gamma_n \Vdash_{\mathrm{I}_u} u : \mu; \tag{5.17}$$

$$\phi, a; \Phi, \mathrm{H} \leqslant \mathbf{0}, a < \mathrm{H}_i \Vdash_{\mathrm{J}_{c_i}} c_i : \gamma_i; \tag{5.18}$$

$$\phi; \Phi, \mathbf{1} \leqslant \mathrm{H}; x_1 : [a < \mathrm{L}_1] \cdot \delta_1, \ldots, x_n : [a < \mathrm{L}_n] \cdot \delta_n \Vdash_{\mathrm{I}_v} v : \mu; \tag{5.19}$$

$$\phi, a; \Phi, \mathbf{1} \leqslant \mathrm{H}, a < \mathrm{L}_i \Vdash_{\mathrm{M}_{c_i}} c_i : \delta_i; \tag{5.20}$$

where

$$\phi; \Phi, \mathrm{H} \leqslant \mathbf{0} \models \mathrm{I}_{(u,v,\rho)} = \mathrm{I}_u + \mathrm{H}_1 + \ldots + \mathrm{H}_n + \sum_{a<\mathrm{H}_1} \mathrm{J}_{c_1} + \ldots + \sum_{a<\mathrm{H}_n} \mathrm{J}_{c_n};$$

$$\phi; \Phi, \mathbf{1} \leqslant \mathrm{H} \models \mathrm{I}_{(u,v,\rho)} = \mathrm{I}_v + \mathrm{L}_1 + \ldots + \mathrm{L}_n + \sum_{a<\mathrm{L}_1} \mathrm{M}_{c_1} + \ldots + \sum_{a<\mathrm{L}_n} \mathrm{M}_{c_n}.$$

Please notice how the type derivations for (5.12), (5.18) and (5.20) are structurally identical, i.e., their PCF counterparts are the same. Now, let us build index terms $\mathrm{N}_1, \ldots, \mathrm{N}_n$, $\mathrm{P}_{c_1}, \ldots, \mathrm{P}_{c_n}$, I_{uv} and types $\eta_1, \ldots, \eta_n$ such that:

$$\phi; \Phi, \mathrm{H} \leqslant \mathbf{0} \models \mathrm{N}_i = \mathrm{H}_i;$$
$$\phi; \Phi, \mathbf{1} \leqslant \mathrm{H} \models \mathrm{N}_i = \mathrm{L}_i;$$
$$\phi; \Phi, \mathrm{H} \leqslant \mathbf{0} \models \mathrm{I}_{uv} = \mathrm{I}_u;$$
$$\phi; \Phi, \mathbf{1} \leqslant \mathrm{H} \models \mathrm{I}_{uv} = \mathrm{I}_v;$$
$$\phi; \Phi, a < \mathrm{K}_i \models \mathrm{P}_{c_i} = \mathrm{I}_{c_i};$$
$$\phi; \Phi, \mathrm{H} \leqslant \mathbf{0}, a < \mathrm{H}_i \models \mathrm{P}_{c_i}\{a + \mathrm{K}_i/a\} = \mathrm{J}_{c_i};$$
$$\phi; \Phi, \mathbf{1} \leqslant \mathrm{H}, a < \mathrm{L}_i \models \mathrm{P}_{c_i}\{a + \mathrm{K}_i/a\} = \mathrm{M}_{c_i};$$
$$\phi; \Phi, a < \mathrm{K}_i \Vdash \eta_i \cong \tau_i;$$
$$\phi; \Phi, \mathrm{H} \leqslant \mathbf{0}, a < \mathrm{H}_i \Vdash \eta_i\{a + \mathrm{K}_i/a\} \cong \gamma_i;$$
$$\phi; \Phi, \mathbf{1} \leqslant \mathrm{H}, a < \mathrm{L}_i \Vdash \eta_i\{a + \mathrm{K}_i/a\} \cong \delta_i.$$

As a consequence, one can rewrite (5.11), (5.17) and (5.19) as follows:

$$\phi;\Phi;x_1:[a<\mathrm{K}_1]\cdot\eta_1,\ldots,x_n:[a<\mathrm{K}_n]\cdot\eta_n \Vdash_{\mathrm{I}_w} w:\mathtt{Nat}[\mathrm{H}];$$
$$\phi;\Phi,\mathrm{H}\leqslant\mathbf{0};x_1:[a<\mathrm{N}_1]\cdot\eta_1\{a+\mathrm{K}_1/a\},\ldots,x_n:[a<\mathrm{N}_n]\cdot\eta_n\{a+\mathrm{K}_n/a\} \Vdash_{\mathrm{I}_{uv}} u:\mu;$$
$$\phi;\Phi,\mathbf{1}\leqslant\mathrm{H};x_1:[a<\mathrm{N}_1]\cdot\eta_1\{a+\mathrm{K}_1/a\},\ldots,x_n:[a<\mathrm{N}_n]\cdot\eta_n\{a+\mathrm{K}_n/a\} \Vdash_{\mathrm{I}_{uv}} v:\mu;$$

from which one obtains

$$\phi;\Phi;x_1:[a<\mathrm{K}_1+\mathrm{N}_1]\cdot\eta_1,\ldots,x_n:[a<\mathrm{K}_n+\mathrm{N}_n]\cdot\eta_n \Vdash_{\mathrm{I}_w+\mathrm{I}_{uv}} \mathtt{ifz}\ w\ \mathtt{then}\ u\ \mathtt{else}\ v:\mu.$$

Similarly, one obtains that

$$\phi,a;\Phi,a<\mathrm{K}_i+\mathrm{N}_i \Vdash_{\mathrm{P}_{c_i}} c_i:\eta_i;$$

and, as a consequence, that $\phi;\Phi\Vdash_{\mathrm{I}_C} C:\sigma$, where

$$\mathrm{I}_C\equiv\mathrm{I}_w+\mathrm{I}_{uv}+\mathrm{K}_1+\mathrm{N}_1+\ldots+\mathrm{K}_n+\mathrm{N}_n+\sum_{a<\mathrm{K}_1+\mathrm{N}_1}\mathrm{P}_{c_1}+\ldots+\sum_{a<\mathrm{K}_n+\mathrm{N}_n}\mathrm{P}_{c_n}.$$

But observe that

$$\begin{aligned}\phi;\Phi,\mathrm{H}\leqslant\mathbf{0}\models\mathrm{I}_C&=\mathrm{I}_w+\mathrm{I}_u+\mathrm{K}_1+\ldots+\mathrm{K}_n+\sum_{a<\mathrm{K}_1}\mathrm{P}_{c_1}+\ldots+\sum_{a<\mathrm{K}_n}\mathrm{P}_{c_n}\\&\quad+\mathrm{N}_1+\ldots+\mathrm{N}_n+\sum_{a<\mathrm{N}_1}\mathrm{P}\{a+\mathrm{K}_1/a\}+\cdots+\sum_{a<\mathrm{N}_n}\mathrm{P}\{a+\mathrm{K}_n/a\}\\&=\mathrm{I}_w+\mathrm{I}_u+\mathrm{K}_1+\ldots+\mathrm{K}_n+\sum_{a<\mathrm{K}_1}\mathrm{I}_{c_1}+\ldots+\sum_{a<\mathrm{K}_n}\mathrm{I}_{c_n}\\&\quad+\mathrm{H}_1+\ldots+\mathrm{H}_n+\sum_{a<\mathrm{H}_1}\mathrm{J}_{c_1}+\cdots+\sum_{a<\mathrm{H}_n}\mathrm{J}_{c_n}\\&=\mathrm{I}_w+\mathrm{K}_1+\ldots+\mathrm{K}_n+\sum_{a<\mathrm{K}_1}\mathrm{I}_{c_1}+\ldots+\sum_{a<\mathrm{K}_n}\mathrm{I}_{c_n}+\mathrm{I}_{(u,v,\rho)}=\mathrm{I}.\end{aligned}$$

Similarly, one can prove that $\phi;\Phi,\mathbf{1}\leqslant\mathrm{H}\models\mathrm{I}_C=\mathrm{I}$. Summing up, we get $\phi;\Phi\models\mathrm{I}_C=\mathrm{I}$, which is the thesis.

- Consider the case

$$C\equiv(x_m,((t_0,\rho_0),\ldots,(t_n,\rho_n)),\xi)\to(t_m,\rho_m,\xi)\equiv D.$$

By assumption we have that C is typable in PCF and that $\phi;\Phi\Vdash_{\mathrm{I}}D:\sigma$. So, we have that

$$\phi;\Phi\Vdash_{\mathrm{I}_{(t_m,\rho_m)}}(t_m,\rho_m):\tau; \tag{5.21}$$
$$\phi;\Phi\Vdash_{\mathrm{I}_\xi}\xi:(\tau,\sigma); \tag{5.22}$$

where $\phi;\Phi\models\mathrm{I}=\mathrm{I}_{(t_m,\rho_m)}+\mathrm{I}_\xi$. Any closure (t_i,ρ_i) (where $1\leqslant i\leqslant n$ but $i\neq m$) can be typed as follows:

$$\phi;\Phi,a<\mathbf{0}\Vdash_{\mathbf{0}}(t_i,\rho_i):\mu_i$$

for some type μ_i. This is because all these closures are by hypothesis typable in PCF and, moreover, $\Phi,a<\mathbf{0}$ is inconsistent. For obvious reasons,

$$\phi;\Phi,a<\mathbf{1}\Vdash_{\mathrm{I}_{(t_m,\rho_m)}}(t_m,\rho_m):\tau.$$

Finally, we can build the following type derivation

$$\frac{\phi;\Phi \Vdash \tau\{\mathbf{0}/b\} \sqsubseteq \tau}{\phi;\Phi;x_1 : [a_1 < \mathbf{0}]\cdot\mu_1,\ldots,x_m : [a < \mathbf{1}]\cdot\tau,\ldots,x_n : [a_n < \mathbf{0}]\cdot\mu_n \Vdash_{\mathsf{0}} x_m : \tau}$$

But all this implies that $\phi;\Phi \Vdash_{\mathrm{I}_C} C : \sigma$ where $\phi;\Phi \models \mathrm{I}_C = \mathrm{I} + \mathbf{1}$, which implies the thesis.

This concludes the proof. □

Relative completeness for programs is a direct consequence of Weighted Subject Expansion:

Theorem 5.6 (Relative Completeness for Programs)**.** *Let t be a* PCF *program such that $t \Downarrow^n \underline{\mathtt{m}}$. Then, there exist two index terms* I *and* J *such that $[\![\mathrm{I}]\!]^{\mathcal{U}} \leqslant n$ and $[\![\mathrm{J}]\!]^{\mathcal{U}} = m$ and such that the term t is typable in* dℓPCF *as $\vdash^{\mathcal{U}}_{\mathrm{I}} t : \mathtt{Nat}[\mathrm{J}]$.*

Proof. By induction on n using Weighted Subject Expansion and Lemma 4.1. □

5.3. **Uniformization and Relative Completeness for Functions.** It is useful to recall that by *relative completeness for functions* we mean the following: for each PCF term t computing a total function f in time expressed by a function g there exists a type derivation in dℓPCF whose index terms capture both the extensional functional behavior f and the intensional property g. Anticipating on what follows, and using an intuitive notation, this can be expressed by a typing judgement like

$$a;\varnothing;x : \mathtt{Nat}[a] \vdash_{\mathtt{g}(a)} t : \mathtt{Nat}[\mathtt{f}(a)].$$

In order to show this form of relative completeness, a *uniformization* result for type derivations needs to be proved.

Suppose that $\{\pi\}_{n\in\mathbb{N}}$ is a sufficiently "regular" (i.e. recursively enumerable) family of type derivations such that any π_n is mapped by $(\!|\cdot|\!)$ to the *same* PCF type derivation. Uniformization tells us that with the hypothesis above, there is a *single* type derivation π which captures the whole family $\{\pi_n\}_{n\in\mathbb{N}}$. In other words, uniformization is an extreme form of polymorphism. Note that, for instance, uniformization does not hold in intersection types, where *uniform typing* permits only to define small classes of functions [28, 8, 9].

More formally, a family $\{\pi_n\}_{n\in\mathbb{N}}$ of type derivations is said to be *recursively enumerable* if there is a computable function f which, on input n, returns (an encoding of) π_n. Similarly, recursively enumerable families of index terms, types and modal types can be defined.

It is easy to turn "uniform families" of semantic entailments into one compact form:

Lemma 5.7. *1. If for every $n \in \mathbb{N}$ it holds that $\phi;\Phi\{\boldsymbol{n}/a\} \models^{\mathcal{E}} \mathrm{I}\{\boldsymbol{n}/a\} \simeq \mathrm{J}\{\boldsymbol{n}/a\}$, then $\phi, a;\Phi \models^{\mathcal{E}} \mathrm{I} \simeq \mathrm{J}$.*

2. If for every $n \in \mathbb{N}$ it holds that $\phi;\Phi\{\boldsymbol{n}/a\} \models^{\mathcal{E}} \mathrm{I}\{\boldsymbol{n}/a\} \leqslant \mathrm{J}\{\boldsymbol{n}/a\}$, then $\phi, a;\Phi \models^{\mathcal{E}} \mathrm{I} \leqslant \mathrm{J}$.

Proof. This is just an trivial consequence of the way semantic entailment is defined. Suppose, for example, that for every $n \in \mathbb{N}$ the following holds $\phi;\Phi\{\mathbf{n}/a\} \models^{\mathcal{E}} \mathrm{I}\{\mathbf{n}/a\} \simeq \mathrm{J}\{\mathbf{n}/a\}$. Now, what should we do to prove $\phi, a;\Phi \models^{\mathcal{E}} \mathrm{I} \simeq \mathrm{J}$? We should prove that for every value of the variables in ϕ, a satisfying Φ, I and J are equal in the sense of Kleene. But this is just what the hypothesis ensures. □

Before embarking on the proof of uniformization for type derivations, it makes sense to prove the same result for index terms and types, respectively.

Lemma 5.8 (Uniformizing Index Terms)**.** *Suppose that:*
1. $\{\mathrm{I}_n\}_{n\in\mathbb{N}}$ *is recursively enumerable, where for every* $n \in \mathbb{N}$, I_n *is an index term on a signature* $\Sigma_{\mathcal{U}}$*;*
2. *There is a finite set of variables* $\phi = a_1, \ldots, a_m$ *such that any variables appearing in any* I_n *is in* ϕ

Then there is a term I *on the signature* $\Sigma_{\mathcal{U}}$ *such that* $\phi; \varnothing \vdash^{\mathcal{U}} \mathrm{I}\{\boldsymbol{n}/a\} \simeq \mathrm{I}_n$ *for every* n*.*

Proof. Consider the function $f : \mathbb{N}^{m+1} \to \mathbb{N}$ defined as follows:

$$(x_0, x_1, \ldots, x_m) \mapsto [\![\mathrm{I}_{x_0}]\!]^{\mathcal{U}}_{[a_1 \leftarrow x_1, \ldots, a_n \leftarrow x_m]}.$$

An algorithm computing f can be defined as follows:
- From x_0, compute I_{x_0}. Again, this can be done effectively.
- Evaluate I_{x_0} where the variables $a_1, \ldots, a_n$ takes values $x_1, \ldots, x_n$, respectively.

In other words, f is computable. Thus, the existence of a term I like the one required is a consequence of the universality of the equational program $\mathcal{U}$. □

Observe how the index terms in $\{\mathrm{I}_n\}_{n\in\mathbb{N}}$ need not be defined for all values of the variables occurring in them. More: their domains of definition can all be different. The way I is defined, however, ensures that $[\![\mathrm{I}\{\mathbf{n}/a\}]\!]$ is defined iff $[\![\mathrm{I}_n]\!]$ is defined. Uniformizing types requires a little more care:

Lemma 5.9 (Uniformizing Types and Modal Types)**.** *Suppose that* $\{\pi_n\}_{n\in\mathbb{N}}$ *is recursively enumerable and that:*
1. *for every* $n \in \mathbb{N}$, $\pi_n \rhd \phi; \Phi_n \vdash^{\mathcal{U}} \sigma_n \Downarrow$*;*
2. *for every* $n, m \in \mathbb{N}$, $(\!|\sigma_n|\!) = (\!|\sigma_m|\!)$*;*
3. *every* Φ_n *have the form* $\mathrm{I}^n_1 \leqslant \mathrm{J}^n_1, \ldots, \mathrm{I}^n_m \leqslant \mathrm{J}^n_m$*, where* m *does not depend on* n*.*

Then there is one type σ *such that:*
1. $\phi, a; \Phi \vdash^{\mathcal{U}} \sigma \Downarrow$*;*
2. $\Phi = \mathrm{I}_1 \leqslant \mathrm{J}_1, \ldots, \mathrm{I}_m \leqslant \mathrm{J}_m$*;*
3. *for every* $1 \leqslant p \leqslant m$*, both* $\phi; \varnothing \vdash^{\mathcal{U}} \mathrm{I}_p\{\boldsymbol{n}/a\} \simeq \mathrm{I}^n_p$ *and* $\phi; \varnothing \vdash^{\mathcal{U}} \mathrm{J}_p\{\boldsymbol{n}/a\} \simeq \mathrm{J}^n_p$*;*
4. *for every* $n \in \mathbb{N}$*, it holds that* $\phi; \Phi\{\boldsymbol{n}/a\} \vdash^{\mathcal{U}} \sigma\{\boldsymbol{n}/a\} \cong \sigma_n$*.*

Moreover, the same statement holds for modal types.

Proof. The proof goes by induction on the structure of the type $(\!|\sigma_0|\!)$ and of the modal type $(\!|A_0|\!)$. An essential ingredient in the proof is, of course, Lemma 5.8. Suppose, as an example, that $(\!|\sigma_0|\!) \equiv \mathtt{Nat}$. This implies that there are index terms $\mathrm{K}_n, \mathrm{H}_n$ such that, for every $n \in \mathbb{N}$,

$$\sigma_n \equiv \mathtt{Nat}[\mathrm{K}_n, \mathrm{H}_n].$$

Now, let $\mathrm{I}_1, \mathrm{J}_1, \ldots, \mathrm{I}_m, \mathrm{J}_m, \mathrm{K}, \mathrm{H}$ be the index terms obtained from the families

$$\{\mathrm{I}^n_1\}_{n\in\mathbb{N}}, \{\mathrm{J}^n_1\}_{n\in\mathbb{N}}, \ldots, \{\mathrm{I}^n_m\}_{n\in\mathbb{N}}, \{\mathrm{J}^n_m\}_{n\in\mathbb{N}}, \{\mathrm{K}_n\}_{n\in\mathbb{N}}, \{\mathrm{H}_n\}_{n\in\mathbb{N}}$$

through Lemma 5.8. Let Φ be just $\mathrm{I}_1 \leqslant \mathrm{J}_1, \ldots, \mathrm{I}_m \leqslant \mathrm{J}_m$ and let σ be $\mathtt{Nat}[\mathrm{K}, \mathrm{H}]$. From $\pi_n \rhd \phi; \Phi_n \vdash^{\mathcal{U}} \sigma_n \Downarrow$, it follows that

$$\phi; \Phi\{\mathbf{n}/a\} \models^{\mathcal{U}} \mathrm{K}\{\mathbf{n}/a\} \Downarrow; \tag{5.23}$$

$$\phi; \Phi\{\mathbf{n}/a\} \models^{\mathcal{U}} \mathrm{H}\{\mathbf{n}/a\} \Downarrow . \tag{5.24}$$

By Lemma 5.7, it follows that

$$\phi, a; \Phi \models^{\mathcal{U}} \mathrm{K} \Downarrow;$$
$$\phi, a; \Phi \models^{\mathcal{U}} \mathrm{H} \Downarrow;$$

which implies $\phi, a; \Phi \vdash^{\mathcal{U}} \sigma \Downarrow$. From (5.23) and $\phi; \varnothing \vdash^{\mathcal{U}} \mathrm{K}\{\mathbf{n}/a\} \simeq \mathrm{K}_n$, it follows that

$$\phi; \Phi\{\mathbf{n}/a\} \models^{\mathcal{U}} \mathrm{K}\{\mathbf{n}/a\} = \mathrm{K}_n.$$

Similarly, from 5.24 one obtains

$$\phi; \Phi\{\mathbf{n}/a\} \models^{\mathcal{U}} \mathrm{H}\{\mathbf{n}/a\} = \mathrm{H}_n.$$

As a consequence, $\phi; \Phi\{\mathbf{n}/a\} \vdash^{\mathcal{U}} \sigma\{\mathbf{n}/a\} \cong \sigma_n$. □

Now that we are able to unify a denumerable family of types into one, we have all the necessary tools to turn a family of judgements into one. For *subtyping* judgments, the task is relatively simple, because types and index terms occurring inside any subtyping derivation also occur in its conclusion:

Lemma 5.10 (Uniformizing Subtyping Judgments)**.** *If for every $n \in \mathbb{N}$ it holds that $\phi; \Phi\{\boldsymbol{n}/a\} \vdash^{\mathcal{E}} \sigma\{\boldsymbol{n}/a\} \sqsubseteq \tau\{\boldsymbol{n}/a\}$, then $\phi, a; \Phi \vdash^{\mathcal{E}} \sigma \sqsubseteq \tau$.*

Proof. This is an induction on the structure of a proof of σ. If, as an example, $\sigma \equiv \mathtt{Nat}[\mathrm{I}, \mathrm{J}]$, then $\tau \equiv \mathtt{Nat}[\mathrm{K}, \mathrm{H}]$. From the hypothesis, we know that

$$\phi; \Phi\{\mathbf{n}/a\} \vdash^{\mathcal{E}} \mathrm{K}\{\mathbf{n}/a\} \leqslant \mathrm{I}\{\mathbf{n}/a\};$$
$$\phi; \Phi\{\mathbf{n}/a\} \vdash^{\mathcal{E}} \mathrm{J}\{\mathbf{n}/a\} \leqslant \mathrm{H}\{\mathbf{n}/a\}.$$

By Lemma 5.7, we can conclude that

$$\phi; \Phi \vdash^{\mathcal{E}} \mathrm{K} \leqslant \mathrm{I};$$
$$\phi; \Phi \vdash^{\mathcal{E}} \mathrm{J} \leqslant \mathrm{H};$$

which immediately yields the thesis. □

In *typing* judgments, on the other hand, there can be types and index terms which occur in the derivation, but not in its conclusion — think about how applications are typed. We then need to impose some further constraints on the kind of (type derivation) families which we can unify:

Lemma 5.11 (Uniformizing Typing Judgments)**.** *If for every $n \in \mathbb{N}$ it holds that $\pi_n \rhd \phi; \Phi\{\boldsymbol{n}/a\}; \Gamma\{\boldsymbol{n}/a\} \vdash^{\mathcal{U}}_{\mathrm{I}\{\boldsymbol{n}/a\}} t : \sigma\{\boldsymbol{n}/a\}$, where $\{\pi_n\}_{n\in\mathbb{N}}$ is recursively enumerable and such that $(\!|\pi_n|\!) = (\!|\pi_m|\!)$ for every $n, m \in \mathbb{N}$, then $\phi, a; \Phi; \Gamma \vdash^{\mathcal{U}}_{\mathrm{I}} t : \sigma$.*

Proof. The proof goes by induction on the structure of t. Some interesting cases:

- Suppose that t is a variable x. Then π_n has the following shape:

$$\frac{\begin{array}{c}\phi; \Phi\{\mathbf{n}/a\} \models^{\mathcal{U}} \mathbf{0} \leqslant \mathrm{J}\{\mathbf{n}/a\} \qquad \phi; \Phi\{\mathbf{n}/a\} \models^{\mathcal{U}} \mathbf{1} \leqslant \mathrm{I}\{\mathbf{n}/a\} \\ \phi; \Phi\{\mathbf{n}/a\} \vdash^{\mathcal{U}} \sigma\{\mathbf{n}/a\}\{\mathbf{0}/b\} \sqsubseteq \tau\{\mathbf{n}/a\} \\ \phi; \Phi\{\mathbf{n}/a\} \vdash^{\mathcal{U}} ([a < \mathrm{I}\{\mathbf{n}/a\}] \cdot \sigma) \Downarrow \qquad \phi; \Phi\{\mathbf{n}/a\} \vdash^{\mathcal{U}} \Delta\{\mathbf{n}/a\} \Downarrow\end{array}}{\phi; \Phi\{\mathbf{n}/a\}; \Delta\{\mathbf{n}/a\}, x : [b < \mathrm{I}\{\mathbf{n}/a\}] \cdot \sigma\{\mathbf{n}/a\} \vdash^{\mathcal{U}}_{\mathrm{J}\{\mathbf{n}/a\}} x : \tau\{\mathbf{n}/a\}}\ V$$

Notice that $\sigma\{\mathbf{n}/a\}\{\mathbf{0}/b\}$ is literally the same as $\sigma\{\mathbf{0}/b\}\{\mathbf{n}/a\}$. Lemma 5.7 and Lemma 5.11 allow us to derive the following

$$\phi, a; \Phi \models^{\mathcal{U}} \mathbf{0} \leqslant \mathrm{J};$$
$$\phi, a; \Phi \models^{\mathcal{U}} \mathbf{1} \leqslant \mathrm{I};$$
$$\phi, a; \Phi \vdash^{\mathcal{U}} \sigma\{\mathbf{0}/b\} \sqsubseteq \tau;$$
$$\phi, a; \Phi \vdash^{\mathcal{U}} ([a < \mathrm{I}] \cdot \sigma) \Downarrow;$$
$$\phi, a; \Phi \vdash^{\mathcal{U}} \Delta \Downarrow;$$

from which the thesis easily follows.

- Suppose that t is uv. Then the derivations in $\{\pi_n\}_{n\in\mathbb{N}}$ have the following shape:

$$\dfrac{\begin{array}{c}\phi; \Phi\{\mathbf{n}/a\}; \Gamma_n \vdash^{\mathcal{U}}_{\mathrm{J}_n} t : [b < \mathrm{I}_n] \cdot \sigma_n \multimap \tau\{\mathbf{n}/a\} \\ \phi, b; \Phi\{\mathbf{n}/a\}, b < \mathrm{I}_n; \Delta_n \vdash^{\mathcal{U}}_{\mathrm{K}_n} u : \sigma_n \\ \phi; \Phi\{\mathbf{n}/a\} \vdash^{\mathcal{U}} \Sigma\{\mathbf{n}/a\} \sqsubseteq \Gamma_n \uplus \sum_{b<\mathrm{I}_n} \Delta_n \\ \phi; \Phi\{\mathbf{n}/a\} \models^{\mathcal{U}} \mathrm{H}\{\mathbf{n}/a\} \geqslant \mathrm{J}_n + \mathrm{I}_n + \sum_{b<\mathrm{I}_n} \mathrm{K}_n\end{array}}{\phi; \Phi\{\mathbf{n}/a\}; \Sigma\{\mathbf{n}/a\} \vdash^{\mathcal{U}}_{\mathrm{H}\{\mathbf{n}/a\}} tu : \tau\{\mathbf{n}/a\}} \; A$$

By Lemma 5.8 and Lemma 5.9, there are index terms $\mathrm{I}, \mathrm{J}, \mathrm{K}$ and a type σ, and typing contexts Γ and Δ such that the following holds:

$$\phi; \varnothing \models^{\mathcal{U}} \mathrm{I}\{\mathbf{n}/a\} \simeq \mathrm{I}_n;$$
$$\phi; \varnothing \models^{\mathcal{U}} \mathrm{J}\{\mathbf{n}/a\} \simeq \mathrm{J}_n;$$
$$\phi, b; b < \mathrm{I}\{\mathbf{n}/a\} \models^{\mathcal{U}} \mathrm{K}\{\mathbf{n}/a\} \simeq \mathrm{K}_n;$$
$$\phi, b; \Phi, b < \mathrm{I} \vdash^{\mathcal{U}} \sigma \Downarrow;$$
$$\phi, b; \Phi\{\mathbf{n}/a\}, b < \mathrm{I}\{\mathbf{n}/a\} \vdash^{\mathcal{U}} \sigma\{\mathbf{n}/a\} \cong \sigma_n;$$
$$\phi, b; \Phi, b < \mathrm{I} \vdash^{\mathcal{U}} \Gamma \Downarrow;$$
$$\phi, b; \Phi\{\mathbf{n}/a\}, b < \mathrm{I}\{\mathbf{n}/a\} \vdash^{\mathcal{U}} \Gamma\{\mathbf{n}/a\} \cong \Gamma_n;$$
$$\phi, b; \Phi, b < \mathrm{I} \vdash^{\mathcal{U}} \Delta \Downarrow;$$
$$\phi, b; \Phi\{\mathbf{n}/a\}, b < \mathrm{I}\{\mathbf{n}/a\} \vdash^{\mathcal{U}} \Delta\{\mathbf{n}/a\} \cong \Delta_n.$$

From the above, we first of all obtain

$$\phi; \Phi\{\mathbf{n}/a\} \models^{\mathcal{U}} \mathrm{H}\{\mathbf{n}/a\} \geqslant \mathrm{J}\{\mathbf{n}/a\} + \mathrm{I}\{\mathbf{n}/a\} + \sum_{b<\mathrm{I}\{\mathbf{n}/a\}} \mathrm{K}\{\mathbf{n}/a\},$$

that by Lemma 5.7 becomes

$$\phi, a; \Phi \models^{\mathcal{U}} \mathrm{H} \geqslant \mathrm{J} + \mathrm{I} + \sum_{b<\mathrm{I}} \mathrm{K}.$$

Analogously, this time through Lemma 5.10, one easily reach

$$\phi, a; \Phi \vdash^{\mathcal{U}} \Sigma \sqsubseteq \Gamma \uplus \sum_{b<\mathrm{I}} \Delta.$$

Again, one can reach

$$\phi; \Phi\{\mathbf{n}/a\}; \Gamma\{\mathbf{n}/a\} \vdash^{\mathcal{U}}_{\mathrm{J}\{\mathbf{n}/a\}} t : [b < \mathrm{I}\{\mathbf{n}/a\}] \cdot \sigma\{\mathbf{n}/a\} \multimap \tau\{\mathbf{n}/a\};$$
$$\phi, b; \Phi\{\mathbf{n}/a\}, b < \mathrm{I}\{\mathbf{n}/a\}; \Delta\{\mathbf{n}/a\} \vdash^{\mathcal{U}}_{\mathrm{K}\{\mathbf{n}/a\}} u : \sigma\{\mathbf{n}/a\};$$

to which one can apply the induction hypothesis. The thesis easily follows. This concludes the proof. □

Uniformization is the key to prove relative completeness for functions from relative completeness for programs:

Theorem 5.12 (Relative Completeness for Functions)**.** *Suppose that t is a* PCF *term such that $\vdash t : \mathtt{Nat} \to \mathtt{Nat}$. Moreover, suppose that there are two (total and computable) functions $f, g : \mathbb{N} \to \mathbb{N}$ such that $t\ \underline{\mathtt{n}} \Downarrow^{g(n)} \underline{\mathtt{f(n)}}$. Then there are terms $\mathrm{I}, \mathrm{J}, \mathrm{K}$ with $[\![\mathrm{I} + \mathrm{J}]\!] \leqslant g$ and $[\![\mathrm{K}]\!] = f$, such that*

$$a; \varnothing; \varnothing \vdash^{\mathcal{U}}_{\mathrm{I}} t : [b < \mathrm{J}] \cdot \mathtt{Nat}[a] \multimap \mathtt{Nat}[\mathrm{K}].$$

Proof. A consequence of relative completeness for programs (Theorem 5.6) and Lemma 5.11. Indeed, a type derivation for $a; \varnothing; \varnothing \vdash_{\mathrm{I}} t : [b < \mathrm{J}] \cdot \mathtt{Nat}[a] \multimap \mathtt{Nat}[\mathrm{K}]$ can be obtained simply by uniformizing all type derivations π_n for programs in the form $t\underline{\mathtt{n}}$. In turn, those type derivations can be built effectively by way of Subject Expansion. □

6. On the Undecidability of Type Checking

As we have seen in the last two sections, dℓPCF is not only sound, but complete: all true typing judgements involving programs can be derived, and this can be indeed lifted to first-order functions, as explained in Section 5.3.

There is a price to pay, however. Checking a type derivation for correctness is undecidable in general, simply because it can rely on semantic assumptions in the form of inequalities between index terms, or on subtyping judgements, which themselves rely on the properties of the underlying equational program $\mathcal{E}$. If $\mathcal{E}$ is sufficiently involved, e.g. if we work with $\mathcal{U}$, there is no hope to find a decidable complete type checking procedure. In this sense, dℓPCF is a non-standard type system.

Indeed, dℓPCF is not actually a type system, but rather a *framework* in which various distinct type systems can be defined. Concrete type systems can be developed along two axes: on the one hand by concretely instantiating $\mathcal{E}$, on the other by choosing specific and sound formal systems for the verification of semantic assumptions. This way sound and possibly decidable type systems can be derived. Even if completeness can only be achieved if $\mathcal{E}$ is universal, soundness holds for every equational program $\mathcal{E}$. Choosing a simple equational program $\mathcal{E}$ results in a (incomplete) type system for which the problem of checking the inequalities can be much easier, if not decidable. And even if $\mathcal{E}$ remains universal, assumptions could be checked using techniques such as abstract interpretation or theorem proving.

By the way, the just described phenomenon is not peculiar to dℓPCF. Unsurprisingly, program logics have similar properties, since the rule

$$\frac{p \Rightarrow r \quad \{r\}P\{s\} \quad s \Rightarrow q}{\{p\}P\{q\}}$$

is part of most relatively complete Hoare-Floyd logics and, of course, the premises $p \Rightarrow r$ and $s \Rightarrow q$ have to be taken semantically for completeness to hold.

7. dℓPCF and Implicit Computational Complexity

One of the original motivations for the studies which lead to the definition of dℓPCF came from Implicit Computational Complexity. There, one aims at giving characterizations of complexity classes which can often be turned into type systems or static analysis methodologies for the verification of resource usage of programs. Historically [24, 29], what prevented most ICC techniques to find concrete applications along this line was their poor expressive power: the class of programs which can be recognized as being efficient by (tools derived from) ICC systems is often very small and does not include programs corresponding to natural, well-known algorithms. This is true despite the fact that ICC systems are *extensionally* complete — they capture complexity classes seen as classes of *functions*. The kind of Intensional Completeness enjoyed by dℓPCF is much stronger: all PCF programs with a certain complexity can be proved to be so by deriving a typing judgement for them.

Of course, dℓPCF is not at all an implicit system: bounds appear everywhere! On the other hand, dℓPCF allows to analyze the time complexity of higher-order functional programs directly, without translating them into low level programs. In other words, dℓPCF can be viewed as an abstract framework where to experiment new implicit computational complexity techniques.

8. Related Work

Other type systems can be proved to satisfy completeness properties similar to the ones enjoyed by dℓPCF.

The first example that comes to mind is the one of intersection types. In intersection type disciplines, the class of strongly and weakly normalizable lambda terms can be captured [16]. Recently, these results have been refined in such a way that the actual complexity of reduction of the underlying term can be read from its type derivation [14, 7]. What intersection types lack is the possibility to analyze the behavior of a functional term in one single type derivation — all function calls must be typed separately [28, 8, 9]. This is in contrast with Theorem 5.12 which gives a unique type derivation for every PCF program computing a total function on the natural numbers.

Another example of type theories which enjoy completeness properties are refinement type theories [17], as shown in [15]. Completeness, however, only holds in a logical sense: any property which is true in all Henkin models can be captured by refinement types. The kind of completeness we obtain here is clearly more operational: the result of evaluating a program and the time complexity of the process can both be read off from its type.

As already mentioned in the Introduction, linear logic has been a great source of inspiration for the authors. Actually, it is not a coincidence that linear logic was a key ingredient in the development of one of the earliest fully-abstract game models for PCF. Indeed, dℓPCF can be seen as a way to internalize history-free game semantics [1] into a type system. And already BLL and QBAL, both precursors of dℓPCF, have been designed being greatly inspired by the geometry of interaction. dℓPCF is a way to study the extreme consequences of this idea, when bounds are not only polynomials but arbitrary first-order total functions on natural numbers.

References

[1] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *I & C*, 163(2):409–470, 2000.

[2] K. R. Apt, F. S. de Boer, and E.-R. Olderog. *Verification of Sequential and Concurrent Programs*. T. in Comp. Sci. Springer-Verlag, 2009.

[3] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[4] P. Baillot, M. Gaboardi, and V. Mogbil. A polytime functional language from light linear logic. In *ESOP*, volume 6012 of *LNCS*, pages 104–124. Springer, 2010.

[5] P. Baillot and K. Terui. Light types for polynomial time computation in lambda calculus. *I & C*, 207(1):41–62, 2009.

[6] G. Barthe, B. Grégoire, and C. Riba. Type-based termination with sized products. In *CSL*, volume 5213 of *LNCS*, pages 493–507. Springer, 2008.

[7] A. Bernadet and S. Lengrand. Complexity of strongly normalising λ-terms via non-idempotent intersection types. In *FOSSACS*, volume 6604 of *LNCS*, pages 88–107. Springer, 2011.

[8] A. Bucciarelli, S. D. Lorenzis, A. Piperno, and I. Salvo. Some computational properties of intersection types. In *LICS*, pages 109–118. IEEE Comp. Soc., 1999.

[9] A. Bucciarelli, A. Piperno, and I. Salvo. Intersection types and lambda-definability. *MSCS*, 13(1):15–53, 2003.

[10] S. A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM J. on Computing*, 7:70–90, 1978.

[11] K. Crary and S. Weirich. Resource bound certification. In *ACM POPL*, pages 184–198, 2000.

[12] U. Dal Lago. Context semantics, linear logic and computational complexity. *ACM TOCL*, 10(4), 2009.

[13] U. Dal Lago and M. Hofmann. Bounded linear logic, revisited. *LMCS*, 6(4), 2010.

[14] D. de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. *CoRR*, abs/0905.4251, 2009.

[15] E. Denney. Refinement types for specification. In *IFIP-PROCOMET*, pages 148–166, 1998.

[16] M. Dezani-Ciancaglini, E. Giovannetti, and U. de' Liguoro. Intersection Types, Lambda-models and Böhm Trees. In *"Theories of Types and Proofs"*, volume 2, pages 45–97. Math. Soc. of Japan, 1998.

[17] T. Freeman and F. Pfenning. Refinement types for ML. In *PLDI*, pages 268–277, 1991.

[18] J.-Y. Girard. Linear logic. *Theor. Comp. Sci.*, 50:1–102, 1987.

[19] J.-Y. Girard, A. Scedrov, and P. Scott. Bounded linear logic. *Theor. Comp. Sci.*, 97(1):1–66, 1992.

[20] B. Grobauer. Cost recurrences for DML programs. In *ICFP*, pages 253–264, 2001.

[21] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. Found. of Comp. Series. MIT Press, 1992.

[22] J. Hoffmann, K. Aehlig, and M. Hofmann. Multivariate Amortized Resource Analysis. In *ACM POPL*, pages 357–370, 2011.

[23] M. Hofmann. Linear types and non-size-increasing polynomial time computation. In *LICS*, pages 464–473. IEEE Comp. Soc., 1999.

[24] M. Hofmann. Programming languages capturing complexity classes. *ACM SIGACT News*, 31:31–42, 2000.

[25] S. Jost, K. Hammond, H.-W. Loid, and M. Hofmann. Static Determination of Quantitative Resource Usage for Higher-Order Programs. In *ACM POPL*, Madrid, Spain, 2010.

[26] N. Kobayashi and C.-H. L. Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *LICS*, pages 179–188. IEEE Comp. Soc., 2009.

[27] J.-L. Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 20(3):199–207, 2007.

[28] D. Leivant. Discrete polymorphism. In *ACM LFP*, pages 288–297. ACM Press, 1990.

[29] J.-Y. Marion. *Complexité implicite des calculs, de la théorie à la pratique*. Habilitation thesis, Université Nancy 2, 2000.

[30] P. Odifreddi. *Classical Recursion Theory: the Theory of Functions and Sets of Natural Numbers*. Number 125 in Studies in Logic and the Foundations of Mathematics. North-Holland, 1989.

[31] G. D. Plotkin. LCF considerd as a programming language. *Theor. Comp. Sci.*, 5:225–255, 1977.

[32] A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE JSAC*, 21(1):5–19, 2003.

[33] D. M. Volpano, C. E. Irvine, and G. Smith. A sound type system for secure flow analysis. *JCS*, 4(2/3):167–188, 1996.

[34] H. Xi. Dependent types for program termination verification. In *LICS*, pages 231–246. IEEE Comp. Soc., 2001.
[35] H. Xi. Dependent ml an approach to practical programming with dependent types. *J. of Funct. Progr.*, 17(2):215–286, 2007.
[36] H. Xi and F. Pfenning. Dependent types in practical programming. In *ACM POPL*, pages 214–227, 1999.